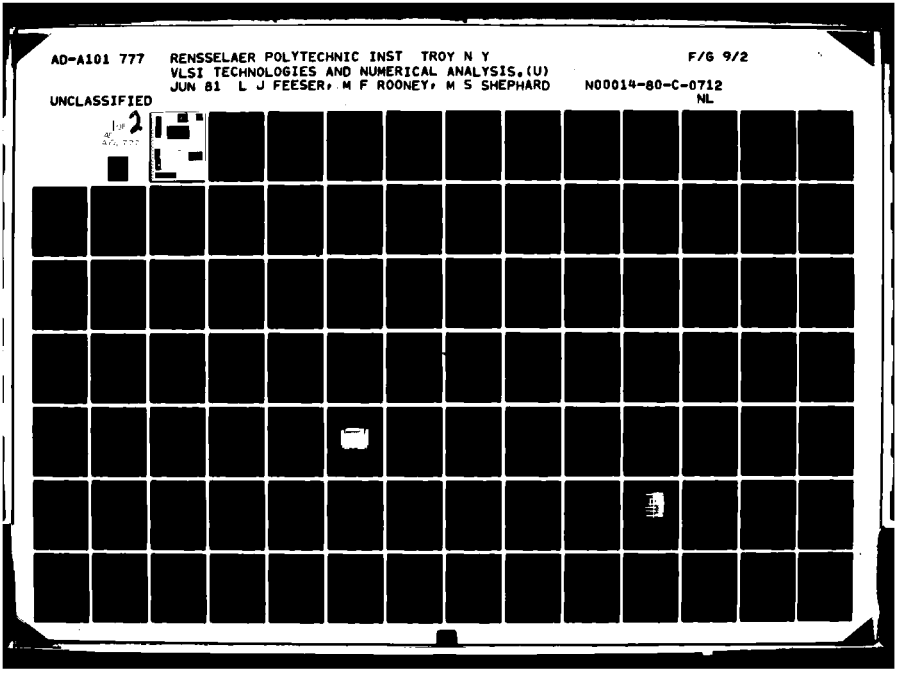
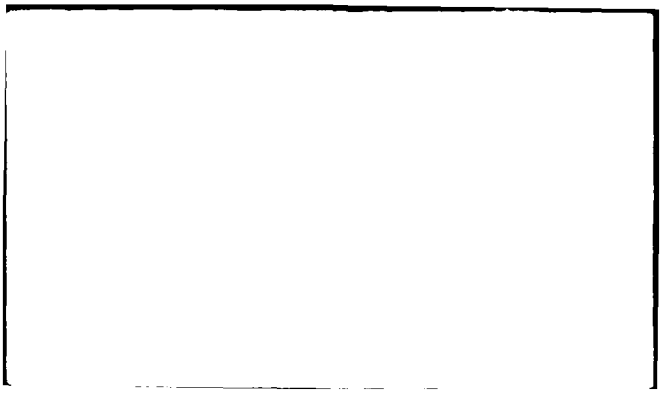
AD-A101 777 RENSSELAER POLYTECHNIC INST TROY N Y F/G 9/2
VLSI TECHNOLOGIES AND NUMERICAL ANALYSIS.(U)
JUN 81 L J FEESER, M F ROONEY, M S SHEPHARD N00014-80-C-0712
UNCLASSIFIED NL





VLSI Technologies and Numerical Analysis

June 29, 1981

Principal Investigators: L. J. Feeser
M. F. Rooney
M. S. Shephard

Agency: Office of Naval Research
Contract No.: N00014-80-C-0712
R. P. I. Project No.: 5-24350

Table of Contents

List of Figures

Introduction

Very large scale integrated circuit technologies (VLSI) are becoming an established technique for implementing computing systems and subsystems. VLSI chips may contain over 100,000 transistors or junctions. These advances have reduced costs and increased the speed of computation, and are now filtering into the domain of structural engineering computation both as general purpose computer improvements and as special processor devices (e.g., 64K memory chips and array processor circuits). These devices may range from programmable read-only memory to, conceivably, Gallagher's finite element pocket computer[7].

Purpose and Scope.

This report will enumerate and expound upon VLSI alternatives available to the numerical analyst involved in structural engineering computations. In addition to showing the potential improvements for the numerical analyst, fundamental constraints of the technologies will be given, thus, documenting the present state of the art in VLSI design. Emphasis will be placed upon indicating availability to the non-circuit designer and predicting the most promising technologies. Further, the vocabulary of VLSI technologies will be tabulated, and will serve as the

cornerstone of future reports.

Developers of numerical analysis code must have a basic understanding of VLSI capabilities in order to efficiently utilize the special features offered by these circuits. In particular, these devices required the extensive use of parallel and pipeline algorithms, yet current numerical methods provide none of the information needed to take advantage of this computer architecture. Although improvements in performance have been achieved by modifying existing codes to run on supercomputers or using array processing peripherals, VLSI technologies combined with parallel pipeline processing schemes will provide much greater increases in speed.

Processor Organization

Reduced processing device costs cause by VLSI implementation have allowed the development of new processor organizations; specifically, organizations where multiple processors are used. Unfortunately, present algorithms are not suited and programmers not trained to take advantage of concurrency in computation. This chapter explores some of the emerging techniques for concurrent processor organization.

Modern Computer.

John von Neumann is credited with developing the concept of data and program stored simultaneously in memory, and further, that programs could be handled the same as data. Most modern computer systems are based upon this notion. Each computer operation consists of six basic steps[14]:

1. Fetch the next instruction from memory.
2. Decode the instruction.
3. Fetch the operands (if any are needed).
4. Perform the decoded instruction upon the operands.
5. Store the results as directed by the instruction.
6. Set up next instruction address and return to fetch the next instruction.

These basic steps are preserved in most VLSI applications, but the scope of each has been extended. (These extensions are examined in more detail in subsequent sections.) An extremely important facet of combined program-data storage is that a program may alter itself during processing. This self-modification ability has been relatively ignored, yet is reappearing in research on finite state machines. (Finite state machines are discussed shortly, and should not be confused with finite element processor.)

There are basically four types of processor organization which are compatible with the von Neumann

concept[4]:

1. SISD - Single instruction single data.
2. SIMD - Single instruction multiple data.
3. MISD - Multiple instruction single data.
4. MIMD - Multiple instruction multiple data.

SISD Machine.

The SISD processing systems comprise the majority of computers in present use. These machines contain one "arithmetic logic unit" (ALU)[24] which handles one piece of data at a time. Each machine operation performs the six basic steps, and produces one result. Each operation must be explicitly performed on each piece of data, yet this apparent inefficiency results in high programming flexibility. This organization, thus, produces good general purpose computer systems. When processor costs (i.e., cost of manufacturing the ALU) are high, as had been the case pre-VLSI, this architecture is considered the optimum for handling large classes of problems. The simplistic power of this architecture should not be underestimiated, however, as many machines are currently being constructed this way (e.g., IBM 370 series, and the DEC VAX family).

A conceptual diagram of an SISD processor is shown in Fig. 1. Input data enters from the left, is operated upon, and the result exits from the right. The wires which carry the binary coded data are designated "data lines" or,

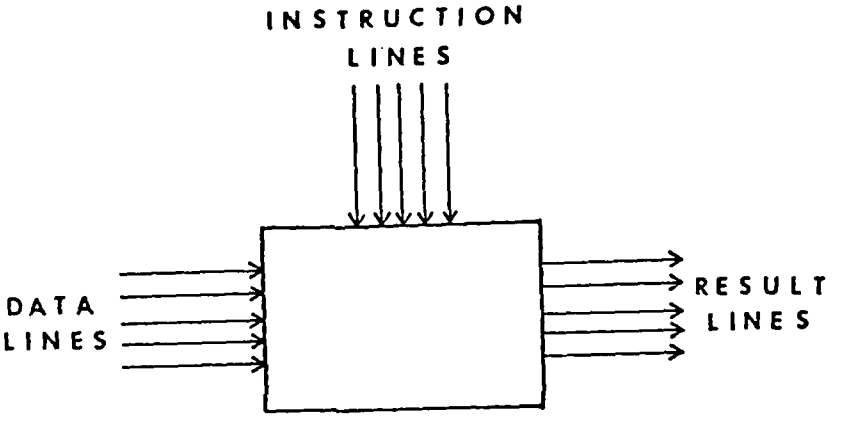


Figure 1. - SISD Organization

collectively, the "data bus". The term "result" is added for clarity, and is not used by circuit designers. Binary coded instructions are shown entering the processor from the top. The wires conveying the instructions are designated as "instruction lines" or "instruction bus". Circuit designers often use the alternate term "control bus".

SIMD Machine.

SIMD processors perform a single instruction on multiple pieces of data simultaneously. In order to do so, as many processing devices are needed as there is data; and further, the instruction signal must be sent to all processors. This technique is generally refered to as "parallel processing".

Fig. 2 illustrates the flow of data and instructions for a SIMD processor. Several pieces of data flow into an equal number of processing devices from the left, are processed simultaneously, and the results are passed out on the right. Once again, encoded instructions are passed to the processing devices from the top, but notice that all devices are connected to the same instruction lines. If all devices are the same, then each will perform the same instruction on its piece of data.

SIMD processors utilize the six basic processor steps, but extend the bandwidth of the data lines (i.e., the number of data items processed at one time). The only changes

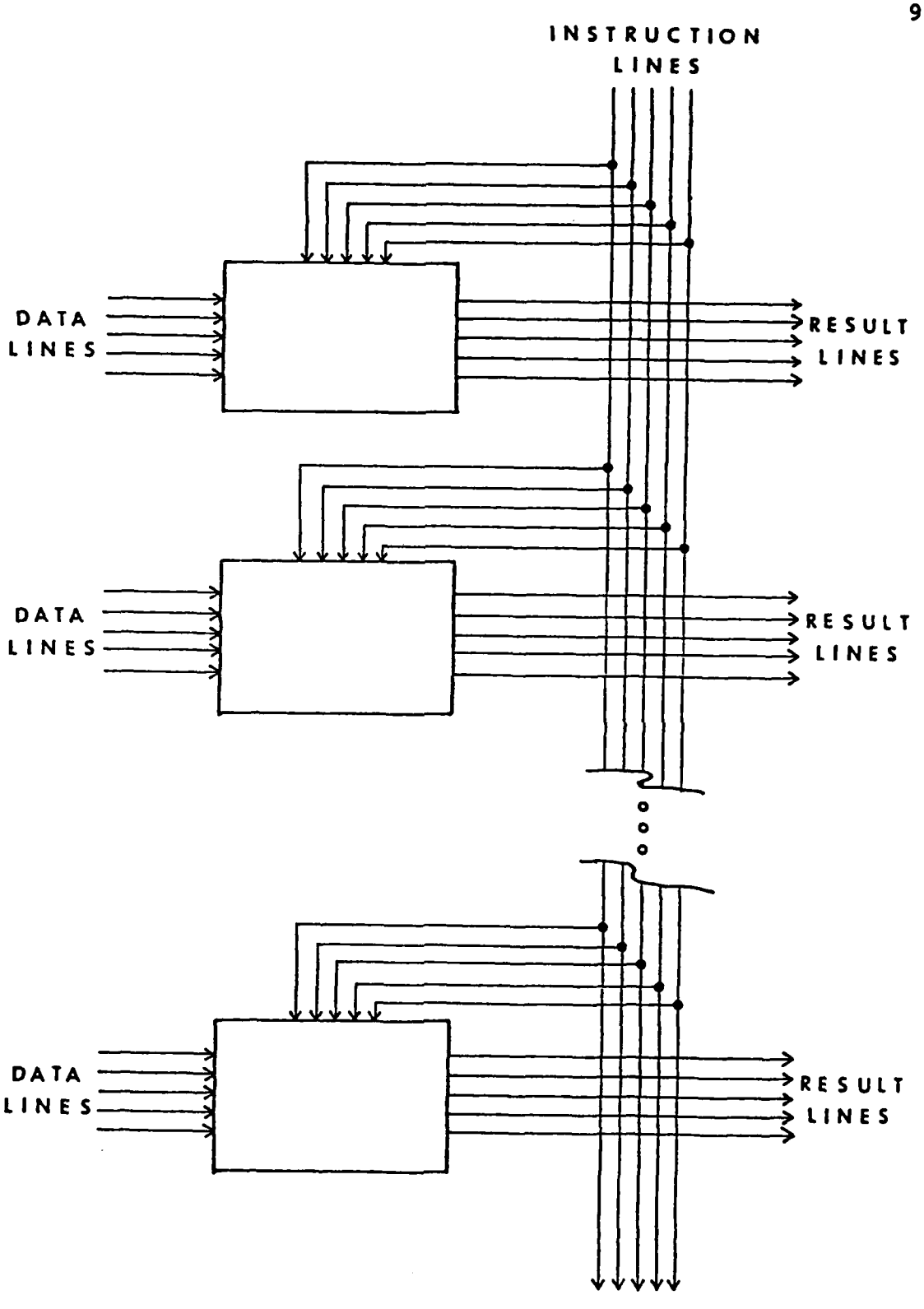


Figure 2 - SIMD Organization

required are in the "fetch operands" and "store result" steps, specifically, the memory must be able to supply the data in block fashion (several pieces at once, called cache memory)[24]. This may be done in several ways. For example, each processing device may have its own connection to the memory, or another processor could load a buffer as shown in Fig. 3. The buffer simply holds the data until needed by the processing device. This requires the memory processor to operate n times faster than the other devices (where n is the bandwidth of the SIMD processor) which is possible for small bandwidths.

A completely different SIMD processor, the hexagonal processor, is shown in Fig. 4. This processing device accepts multiple data items through multiple sets of data lines, and produces a single result. In Fig. 4 data item "A" arrives through the upper left, data item "B" arrives through the lower left, are processed (e.g., are added together), and the result is passed out to the right. This multiple data item entry can be expanded to any number of items. The instructions lines have been omitted for clarity, but could enter through any of the unused sides. An SIMD processor of this sort is not considered to be a parallel processor, but will be shown to be instrumental in construction special purpose structural engineering processors.

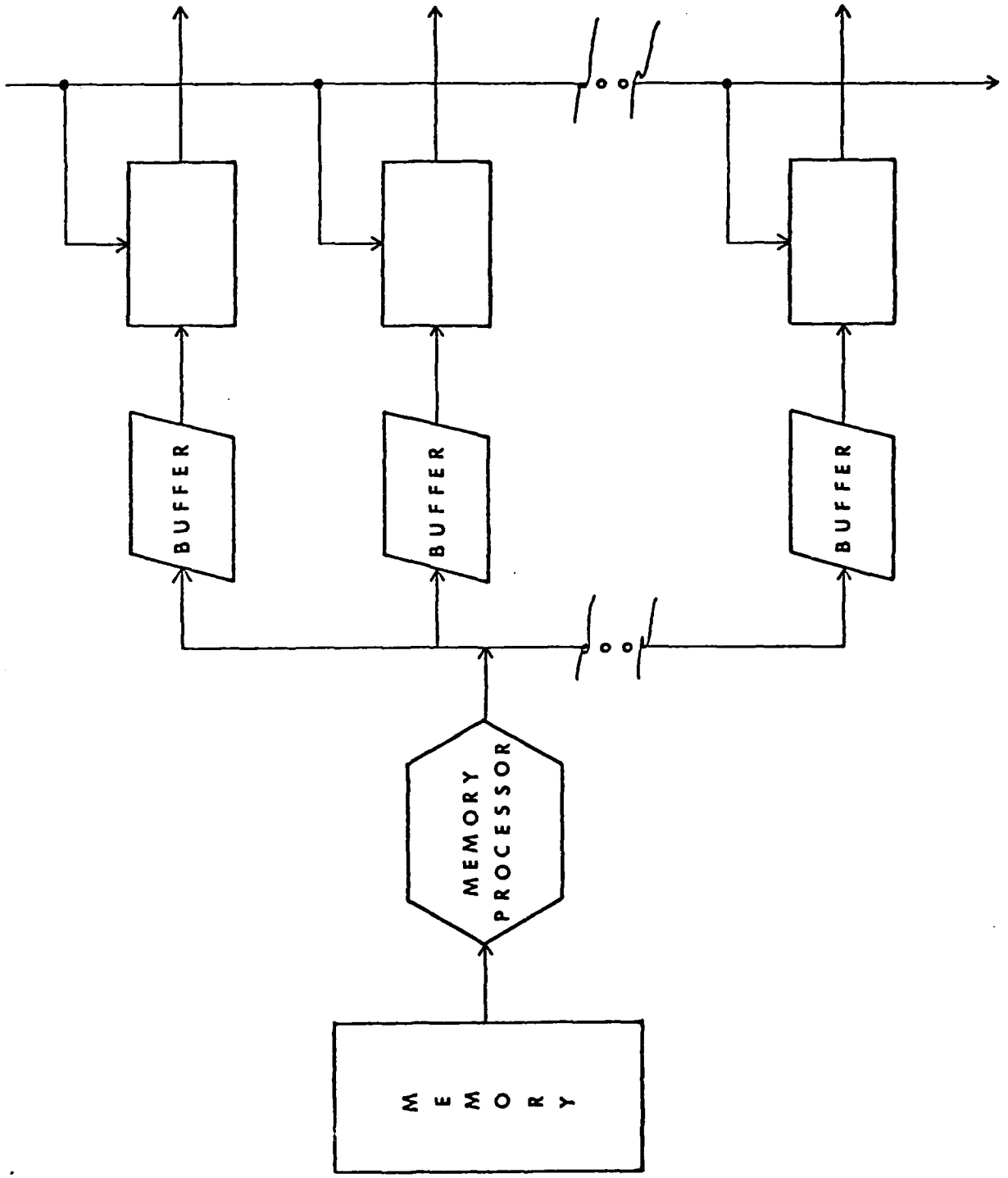


Figure 3 - Buffered Data Transfer

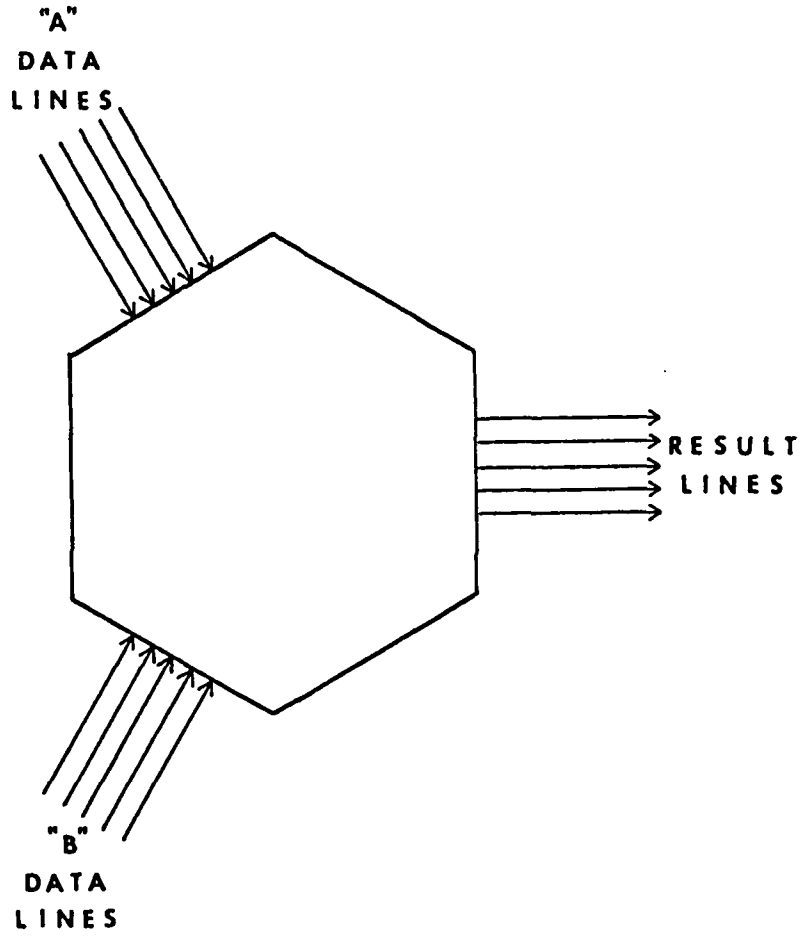


Figure 4 - Hexagonal Processor

MISD Machine.

MISD processors perform multiple instruction on a single piece of data. Obviously, the instructions cannot be performed simultaneously; but rather a sequence of operations is initiated with a single instruction. Once each processor receives data and a constant stream of data flows, the operations are performed simultaneously. A time lag between initiation of the sequence and first results occurs, and is a function of the number and type of processors in the chain. (The timing of operations is considered in a separate section.) This techniques is usually called pipeline processing.

Fig. 5 illustrates an MISD processor arrangement. Data enters on the left and is passed through each processor. The result exits from the right after all operations have been completed. Binary encoded instructions enter from the top. Once a piece of data has entered an MISD processor, it must progress through each processor. Instructions must therefore be sequenced to correspond to the position of the data; this is simple if operation sequence does not change or changes infrequently.

An MISD processor differs from the classic von Newmann machine by increasing the bandwidth of the instructions. Specifically, the "fetch next instruction" and "set-up next instruction" steps require that blocks of instructions be

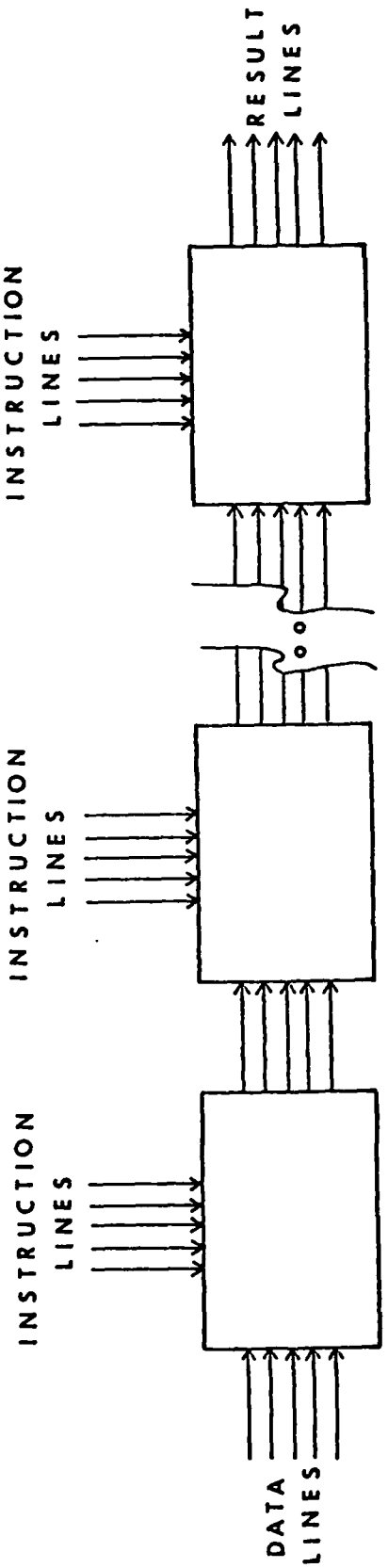


Figure 5 - MISD Organization

loaded a once. A buffered arrangement, similar to data handling in the SIMD machine, is often used; and instructions are stored in blocks of memory.

MIMD Machine.

An MIMD machine performs multiple instructions on multiple data simultaneously. This is accomplished by combining the techniques of MISD and SIMD machines, and is thus called pipelined parallel processing.

Fig. 6 illustrates the combined approach. Sets of data enter on the left, are processed as they progress to the right, and results emerge on the far right. Properly sequenced instructions are fed in from the top, and each column of processing devices is supplied the same instruction.

In the MIMD machine both instruction and data bandwidths are expanded, and fetching from memory becomes a primary concern.

A slightly different configuration arises if parallel and pipeline techniques are applied on a local basis. Consider a pipeline procedure with fixed instructions. The pipeline speed is limited by a the slowest processor. For simplicity, consider the problem of one step taking twice as long as the rest. Fig. 7 demonstrates a solution obtained by localized parallel processing within the pipeline. Data enter on the left and progresses normally down the pipeline

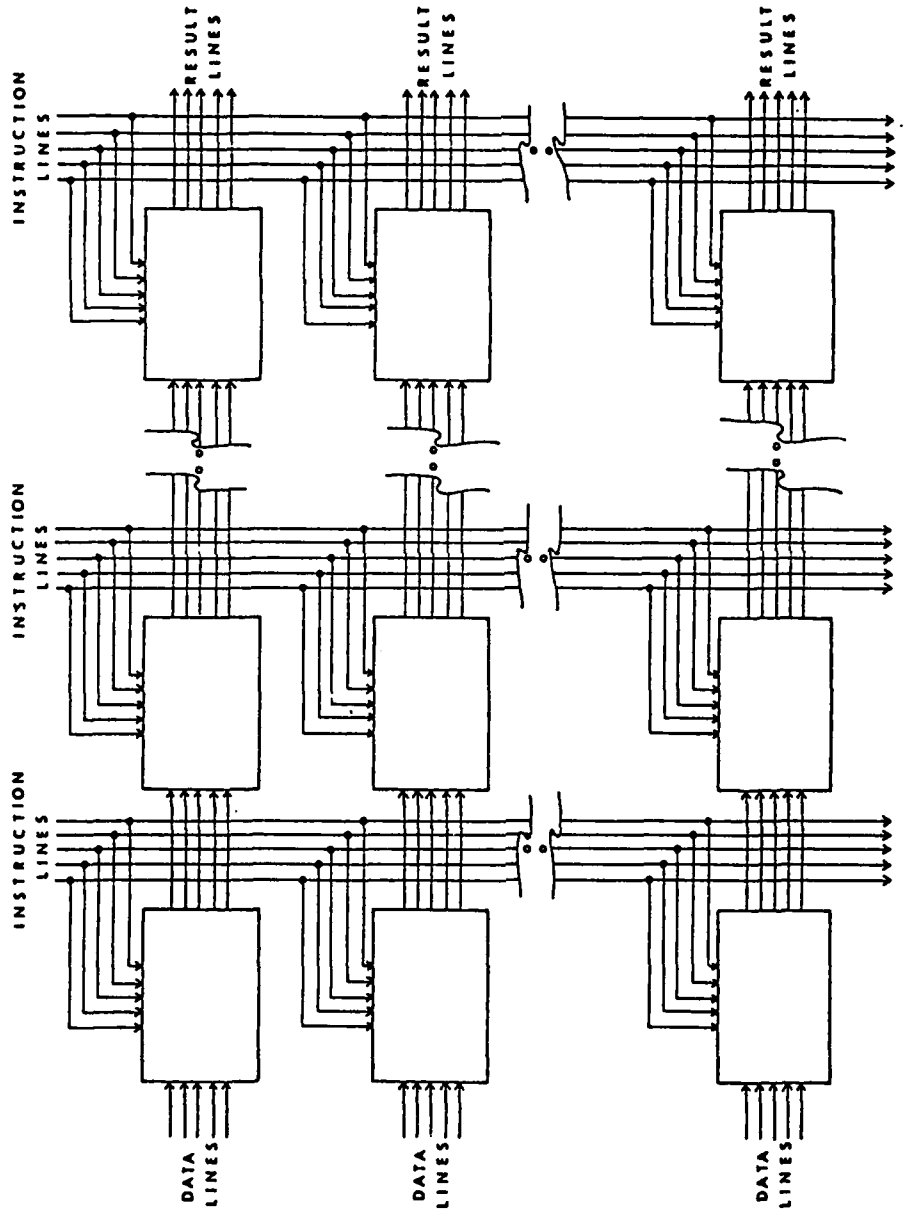


Figure 6 - MIMD Organization

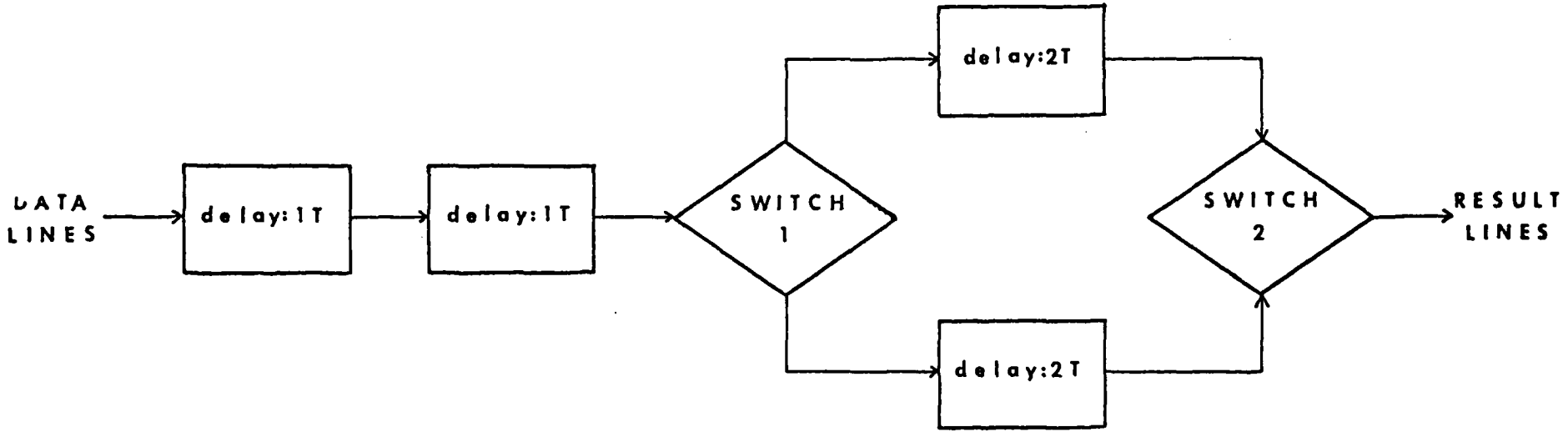


Figure 7 - Localized Parallel Processing

until reaching switch 1. At switch 1 the data is alternately passed to one of the two 2T processors (T represents a fundamental operation time). Thus, at the 2T processor step, two pieces of data are being processed simultaneously, but 1/2 operation out of phase. Switch 2 is inversely linked to switch 1, and alternates to pass the processed data on to subsequent stages. (Timing will be discussed shortly.) This arrangement is technically an MIMD organization, but general purpose extention to VLSI application would be difficult.

Finite State Machine.

A finite state machine is a processing device where the output of the device has an effect upon the instruction or operation being performed. It can be an SISD, MISD, SIMD, or MIMD organization, but each processing device must contain some sort of feedback loop.

Fig. 8 illustrates a simple SISD finite state machine. Data enters on the left, is processed, and results exit on the right. Instructions are fed in from the top. Also, a copy of the results are fed into the top; these wires are called "state lines". The input of the instruction lines are combined with the input of the state lines to determine the operation to be performed.

All lines contain binary signals, and i lines can carry 2^i different codes. If there are i state lines, then the

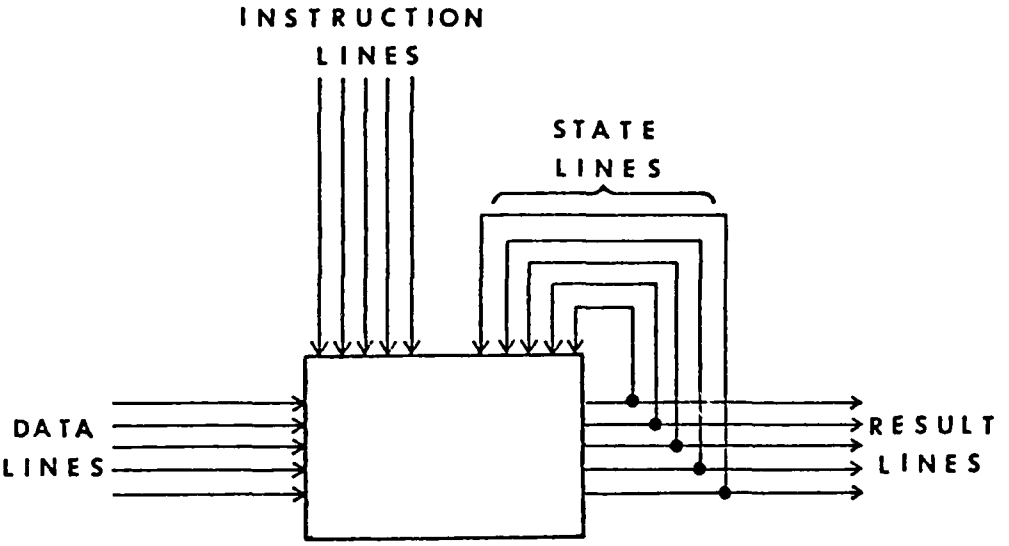


Figure 8 - Finite State Machine

machine may have 2^i states. Because there are a finite number of state lines, the device is known as a finite state machine.

The state lines or rather the feedback loop created by them may be either synchronous or asynchronous. In an asynchronous configuration, the state lines immediately effect the operation being performed; a synchronous configuration only allows the state lines to effect the next computation. Both have applications: asynchronous when a damping operation is required, synchronous when feedback would cause instability. The primary applications in VLSI circuits are for self-modifying programs. Little work has been done with these devices that is of concern to the numerical analyst, but great potential exists.

Binary Processor.

Binary tree processor organization[15] is receiving some consideration lately. The organization consists of a hierarchy of processors, each with two subprocessors. Each processor in the chain, with the exception of the bottom devices, are responsible to subdivide the operation and recombine the results of its subprocessors. The bottom processor do not subdivide, but instead perform the elementary operations.

Fig. 9 shows the organization. Data enters from the left, the task is subdivided and data and subtasks are

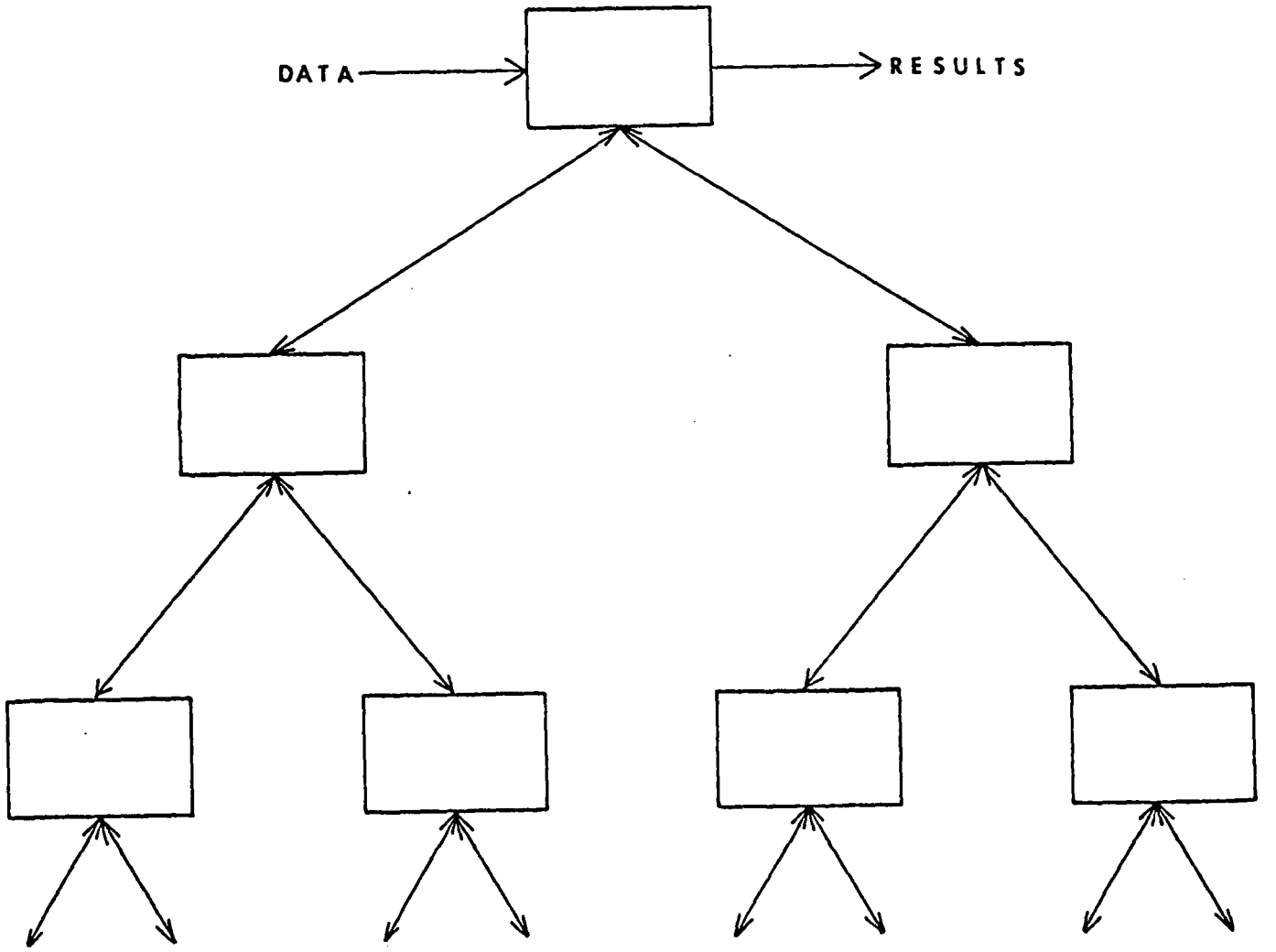


Figure 9 - Binary Tree Organization

passed to subprocessors. Each subprocessor repeats the procedure. When results are obtained at the lowest level, they propagate upwards, and are combined in pairs by each higher level processing device. When the results reach the top, they exit to the right. Each processing device has local sovereignty.

Organizational Difficulties.

Distributing the workload is a major problem when parallel or pipeline architectures are used. If the work (i.e., the delay per processor) is not distributed evenly, then some processors are idle while others are overloaded. This idle time is a waste of processing capabilities and is therefore inefficient. One solution is to combine parallel and pipeline processing at a local level. The flow of instructions become more complicated, but the gains can be quite substantial if the ratios of operation times is large. The problem is further complicated if each processing device performs several operations with different time requirements. The variation in operations prevents localized parallel pipeline improvements. Self-timed arrangements, where each processor signals to the next when the data is ready, helps when a single piece of data is flowing; but is of little value when a stream of data is being processed as the slow processing device backs up the flow.

The nature of VLSI technologies are such that delay times are equal to the time required by electricity to travel a few feet. Communications lines, specifically those connecting processing devices to memory, must be kept short. The capacitance of the communication line also taxes processor capabilities, and also requires short communication lines. This capacitance requirement is critical within the chip itself. As a result, a short regular flow of data within the chip is needed to obtain short delay times. Because von Neumann machines handle programs as data, similar constraints apply to control lines. However, the need for short control lines usually conflicts with short data lines; and a design trade-off must be made. For general purpose machines, an even balence is struck to achieve overall improved performance; yet for individual procedures this middle of the road position is inefficient.

Memory is dynamic in nature, yet the integrity of its contents must be maintained at any given instant. In order to achieve this goal, only the following conditions are allowed to occur:

1. Only one processor may write to a specific memory location at one time.
2. If a location is being written, it may not be read.
3. Several processors may read the same location at

random, but the data may change unexpectedly

or... only one processor may use a memory location, and all others must wait until the first processor releases that location.

For SISD organization, these constraints pose no problems. Other processor organizations will run into interference when several processing devices try to access the same location. If all are allowed to read data, then conflicting results may occur when the fastest processor changes the contents of that location. If others are required to wait, processing speed is decreased, which may be unnecessary if the first processor does not change the contents of memory. A combined solution is theoretically the best, but is not practical to implement without a fixed set of operations.

A proposed solution to access interference and long communication lines is to overlap the "fetch instruction" and "fetch data" steps. That is, while the processing device is computing on the datum, the next instruction could be loaded. However, the number of operands or data items loaded in the "fetch data" step varies with the instruction to be performed. The exact time to load the next instruction is, thus, unclear. In a self-timed system, data and instruction fetched occur randomly due to the independence of processing device operation. This random requests of memory are completely contrary to overlap

techniques. Program branching based upon the results of an operation are made more difficult when fetches are overlapped; particularly in the case of a jump to subroutine where it is imperitive to know where the jump occured from, the loading of the next instruction would destroy that needed information. Once again, for a fixed set of instructions, the procedure has significant merit.

Perhaps the most important factor for the numerical analyst pertaining to extended processor organization is the incompatibility of present algorithms. Present algorithms are based almost exclusively upon SISD processor organization. As a result, they contain the operations to be performed, but give little or no information on the order in which the steps are to be performed. More specifically, no indication of concurrency or interdependency of data is included. Many times several operations could be carried on simultaneously, but the algorithm indicates a prefered order. To further complicate the incompatibility, no consideration is given to the flow of data or instructions. As these new organizations become prevalent, a new scheme for developing algorithms, one that includes data flow and timing concurrency, will evolve.

Advances in Memory Technology.

VLSI technologie demand faster access time from memory circuits. These speed increases are gained through three types of improvements.

VLSI technologies have been applied to memory circuits themselves. This has increased both the speed and density of memory. Other technologies, such as bubble memory[2] have been developed specifically for memory applications. These advances are done by improvements in the hardware.

An obvious method of increasing access time, is to reduce the number of accesses. This is precisely the scheme used in pipeline processing. One access is made to primary memory to obtain the value. This value is then passed along to subsequent processing devices rather than being stored and retrieved by each processor. A similar scheme can be implemented in parallel architectures. The value is obtained with one access, and transmitted simultaneously to all processors which require the value.

The third appoach is to reorganize the storage scheme of memory. Recalling that the length of communication lines is more critical than the number of transistors (and in fact also more costly using VLSI technology), then a hierarchial memory is created, using special purpose processing devices to find the desired value. In its simplest form, the hierarchial memory is a binary tree storage scheme. The

number of memory processors and hence path lengths can then be made proportional to the logarithm of memory size. (Conventional memories are proportional to memory size.) In order to obtain even better performance, memory is grouped by type of access. These types are registers, cache, primary storage (sometimes called core storage), and secondary storage (e.g., disk). Mead and Conway[10] present a typical frequency distribution for these types of accesses:

Type	Frequency
Register	.6
Cache	.38
Primary	.02
Secondary	.000005

Based upon such data, a designer may minimize the total access time by designing short paths to the more frequently used memory. Fig. 3 showed buffers operating as a cache memory.

Speed Increases.

Having briefly explained the extended processor and memory extensions it is time to compare the techniques. Mead and Conway[11] have summarized typical speed-up factors

(as compared to SISD organization) as follows:

Technique	Typical Speed-Up Factor
Memory Hierarchy	10
Pipeline Processing	
Instruction Overlap	2
Special Purpose	n
Parallel Processing	<n

where n denotes the number of processors, and Special Purpose Pipeline Processing refers to the type of machine shown in Fig. 5. For the machine shown in Fig. 6, an increase of slightly less than m * n is possible, where m is the number of processors in a column and n is the number of processing devices in a row.

Timing and Synchronization.

To this point, the sequence of operations has been emphasized; however, the synchronization of operations is ultimately controlled by the time required to perform operations and/or transfer data. The synchronization is further complicated because transistors are not perfect switches, but rather have a distinct time required to charge and discharge output lines. It is absolutely necessary to consider this time behavior, if data is transfered too soon,

a full transition between logic values may not have occured and data will be transfered incorrectly.

The solution to switch delay and waiting for data propagation over long communication lines is timing. Timing may be either synchronous or asynchronous. In a synchronous system, all processing devices operate and transmit data according to an externally supplied signal which is connected to all devices. Transfer of data, both internally and externally, is allowed only during certain times, times when the logic values are known to be stable. Asynchronous operation requires all involved parties to enter a dialog of sorts:

1. The first processor sends a separate signal to indicate the data is available and steady.
2. The second processor removes the data from the lines.
3. The second processor sends a signal to indicate transfer is complete.
4. The first processor may now continue with its other business.

If the first processor does not send the data ready signal until the data is placed upon the lines and the data ready line is as long as the data lines, then it is impossible for the data ready signal to arrive before the data does. Thus, the length of data lines is not important, nor is their delay. Presently, synchronized timing is more common

because of simplified timing design and reduced interprocessor communication.

The circuit(s) which generate the timing signal is called the "clock" and the signal is the "clock signal". Consider the pipeline processor shown in Fig. 10 which is composed of processing devices (rectangles) and switches (triangles). If the devices and switched operated together, the confusion mentioned before would still exist. Rather it is desired that there be two distinct phases: one for processing, and one for transfering data. To accomplish this two clock signals are required, and further that one not operate when the other does. The two signals are designated gamma-1 and gamma-2. During the gamma-1 phase, the data is transfered between processing devices; and during the gamma-2 phase, computations are done. It is common practice to derive the two-phase clock signal from a single clock pulse by counting every other pulse.

Applying VLSI Technologies

The remainder of this report will focus upon how the numerical analyst may apply VLSI technologies to computations relating to structural analysis. There are three broad categories through which VLSI technology will impact: general purpose chips, semi-custom chips, and custom

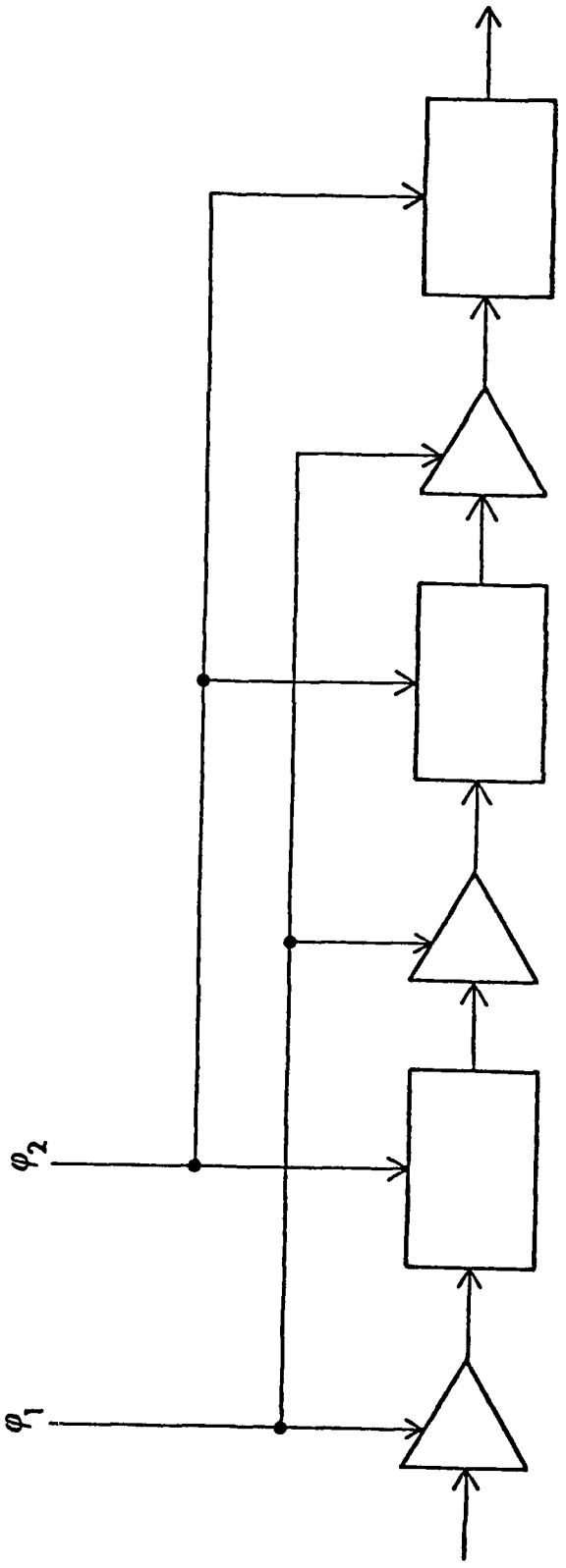


Figure 10 - Clocked Data Transfer

chips.

General Purpose Chips

Perhaps, most of the improvements caused by VLSI technologies will occur with limited involvement of the numerical analyst. Existing architectures and computer lines will evolve toward VLSI implementation from their present small or medium scale integration. The net result will be increases in computation speed and storage capabilities. New architectures will be developed for general purpose computers to further increase speed and capabilities. New types of peripheral devices (e.g., array processors) will become standard equipment. These advances will be developed by the computer manufacturers, without input from the numerical analyst or other special concerns, and as a result, will be improvements to general purpose computation only.

Beyond the normal improvements of existing circuit designs by transfering to VLSI implementations, three new general purpose schemes will emerge: distributed processing, read-only memory, and content addressable memory. The first is a sort of pipeline processing, and will extract the advantages of microcomputers. The latter two will be used in hardware look-up; a scheme where certain values are

pre-calculated and stored, to be used much like mathematic tables were used in pre-calculator times. These techniques are briefly explained below:

Distributed Processor Networks.

Microcomputer power will see the most drastic improvements from VLSI technologies. The low cost of these devices will promote a distributed processing environment. In a distributed processing scheme, the complexity and/or scope of computation is matched to an appropriate size computer. This requires computers to be linked together with a communications network. Generally, a number of small machines are tied to one larger computer; and several of the larger computers are connected to an even larger machine; and so on. As problem size increases, the problem is passed up to the larger machine, and the results are passed back to the smaller machine.

Fig. 11 illustrates a distributed processing arrangement. At the lowest level, the user interfaces to a microcomputer attached to a display. If the microcomputer has insufficient power, the processing is passed to the minicomputer; but because this does not happen continuously, it is possible to connect several users in a similar fashion. This is the classic time sharing procedure except a processor is on both ends. If the minicomputer is insufficient processing proceeds upward to the mainframe.

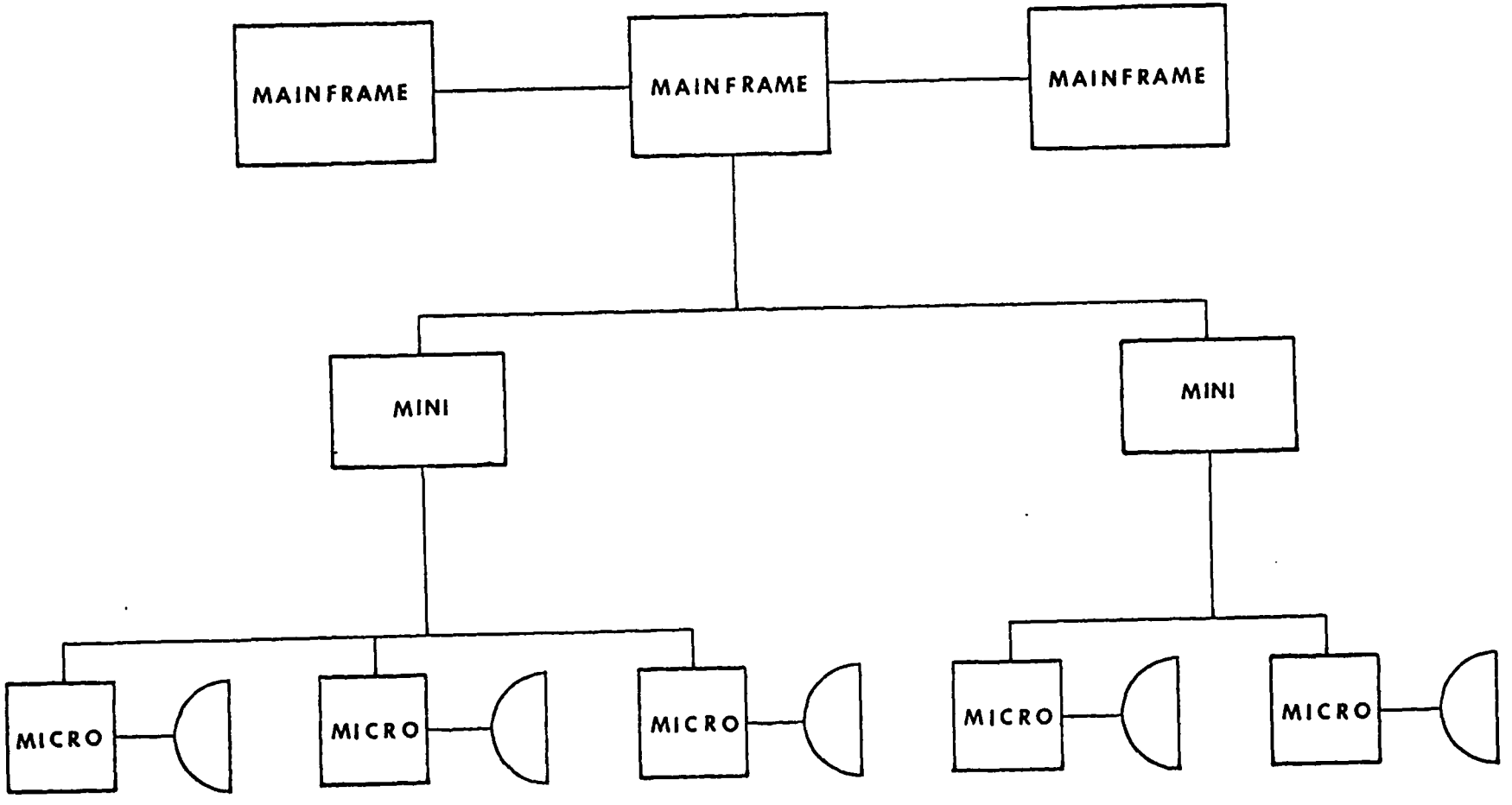


Figure 11 - Distributed Processor Organization

Problems flow upward and data flows downward. In addition, processing may flow latterally to equivalent machines if the processor of desired capability is in use; this lateral capability is shown at the mainframe level in Fig. 11. Should one mainframe be in use, the upward flowing problem is passed latterally to an equivalent machine. This arrangement simplifies the communications prcblems and behaves similar to a binary processor.

Fig. 12 shows an alternate arrangement for a distributed processor. All communications are handled through central manager (a computer), and problems are directly routed to the proper upward level if the microcomputer is unable to handle the problem. This scheme complicates communication, but reduces the amount of information passing. Also note that special I/O (input/output) devices can be readily shared in such an arrangement.

Both distributed processor arrangements require the use of a microcomputer for the man-machine interface and elementary computation. As the power of the microcomputer is increased, less problem passing is done, and more users can be connected to the network.

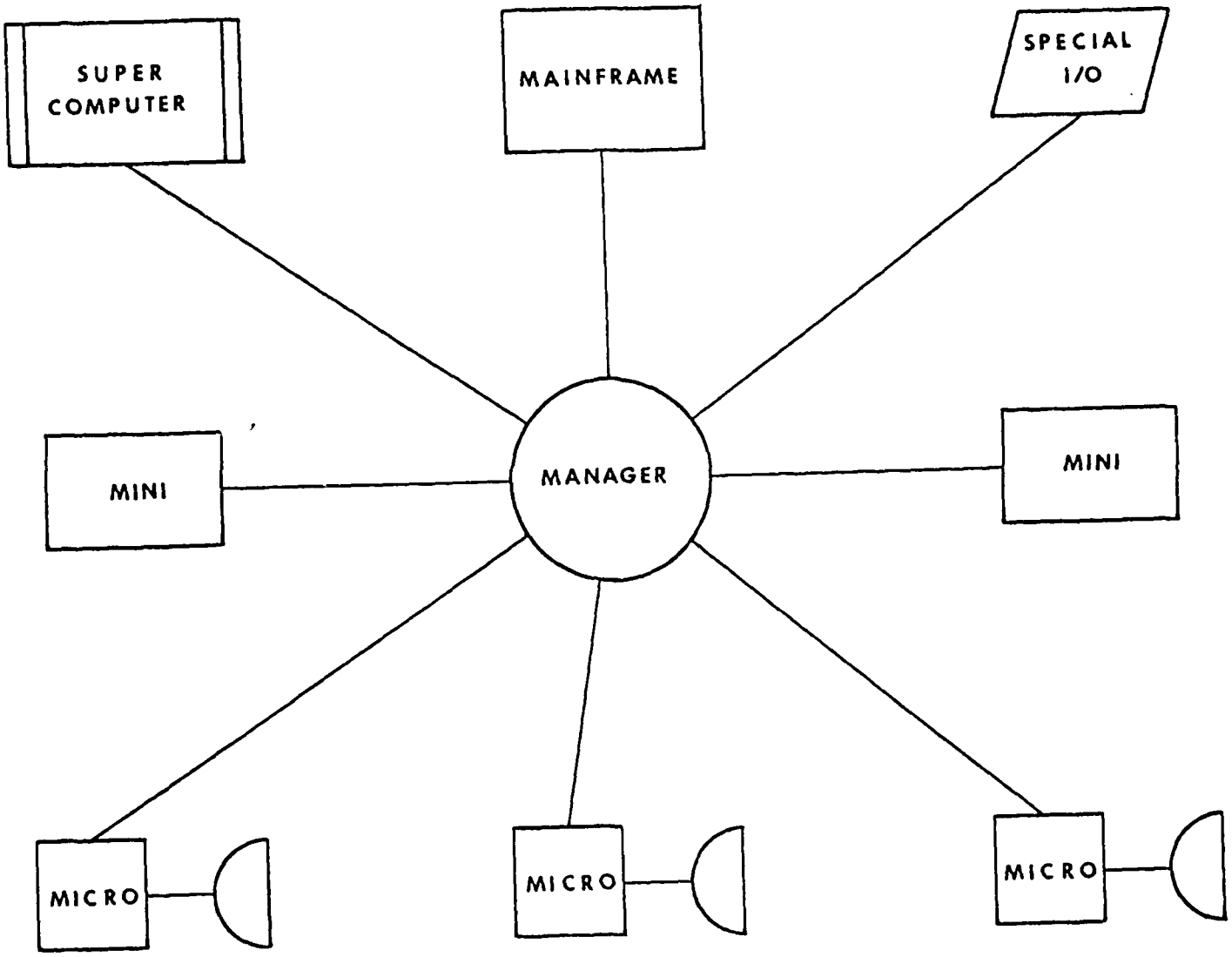


Figure 12 - Alternate Distributed Processor Organization

Read-Only Memory.

Presently, most scientific functions are computed using recursive algorithms on an "as needed" basis. The reduced costs of memory are now providing an alternative called hardware look-up. Fig. 13 illustrates one arrangement for hardware look-up. Data, the value at which the function is to be evaluated, is stored in some memory location called "i". A special processor is connected to that location which constantly samples the data, looks up the corresponding value of the function for the data value, and stores the result value into memory location "i+1".

This arrangement requires that all values of the function be stored, which, for a continuous function (e.g., SIN(X)), is not possible. However, because the input data is binary encoded and a limited number of data lines exist there are only a finite number of possible input values. Specifically, for n data lines, there are 2^n possible input values; and thus, only 2^n values of the functions must exist. For 16 bit input values, 64K of memory is required.

A typical configuration for a read-only memory chip (ROM) is shown in Fig. 14. A binary coded address enters on the left. The address is decoded and closes the appropriate internal switches to allow the contents of the specified memory location to flow out on the right, through the data

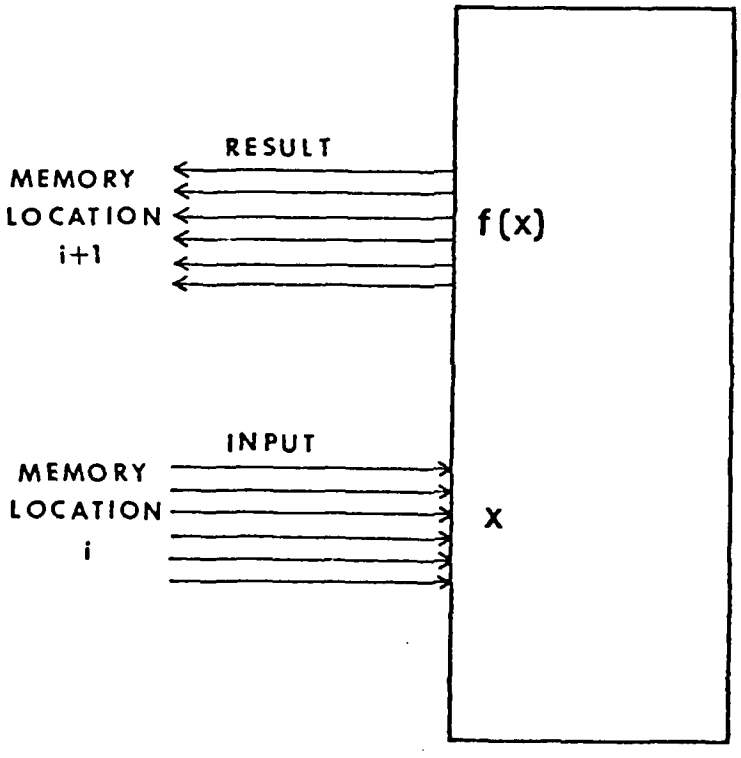


Figure 13 - Hardware Look-Up

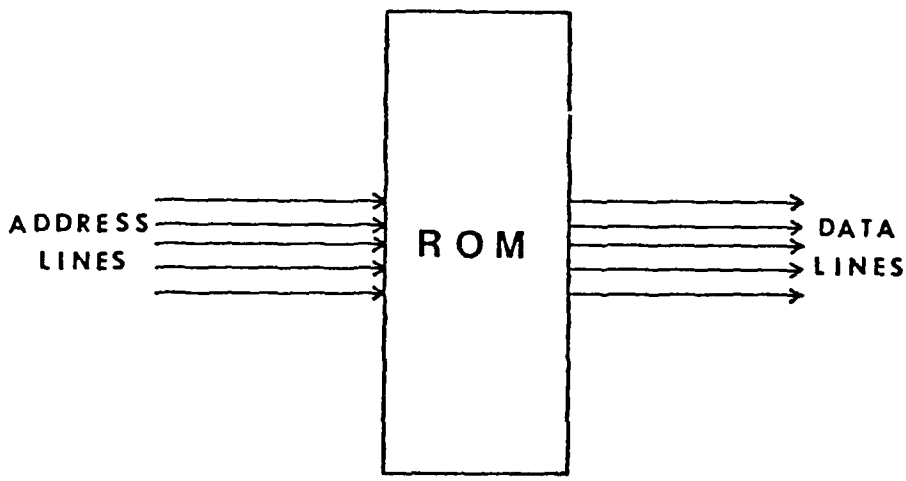


Figure 14 - ROM Chip

lines. The content of each memory location on the ROM is explicitly determined by circuitry, and cannot be changed. As a result, writing into the chip would have no effect; and the chip designated as read-only.

By comparing Figs. 13 and 14, it is evident that the ROM chip can serve as the special processor by connecting the address lines of the ROM to memory location "i" and the ROM data lines to memory location "i".

Fig. 15 illustrates an extended version of ROM based hardware look-up: a function of two variables, f(X,Y). A portion of the address is obtained from memory location "i" and another portion from location "j". The results are stored in location "i+1". Using similar extensions, functions of any number of variables can be developed. By connecting several ROM's to the same memory location for input, several functions can be computed simultaneously as shown in Fig. 16. For example, ROM1 might compute SIN(X), ROM2 might compute COS(X), and ROM3 might compute SQRT(X). Such values might be used in developing a rotation matrix for instance.

The primary difficulty in implementing such a scheme is the cost of developing and manufacturing the ROM chip. Also, until recently, large ROM chips were not available, though several chips could be used to obtain the desired performance. There must be sufficient demand for a

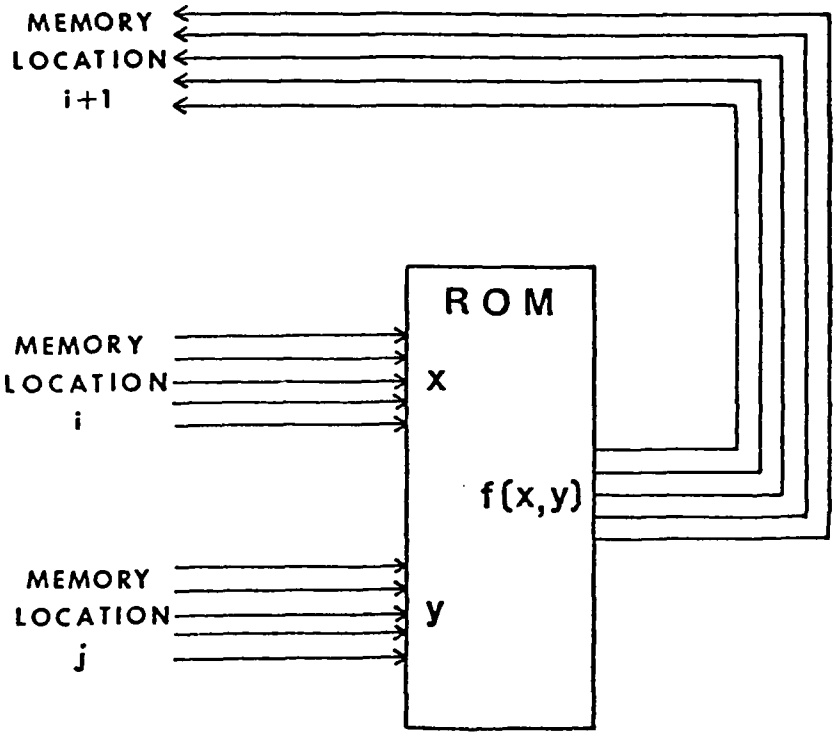


Figure 15 - Extended Hardware Look-Up

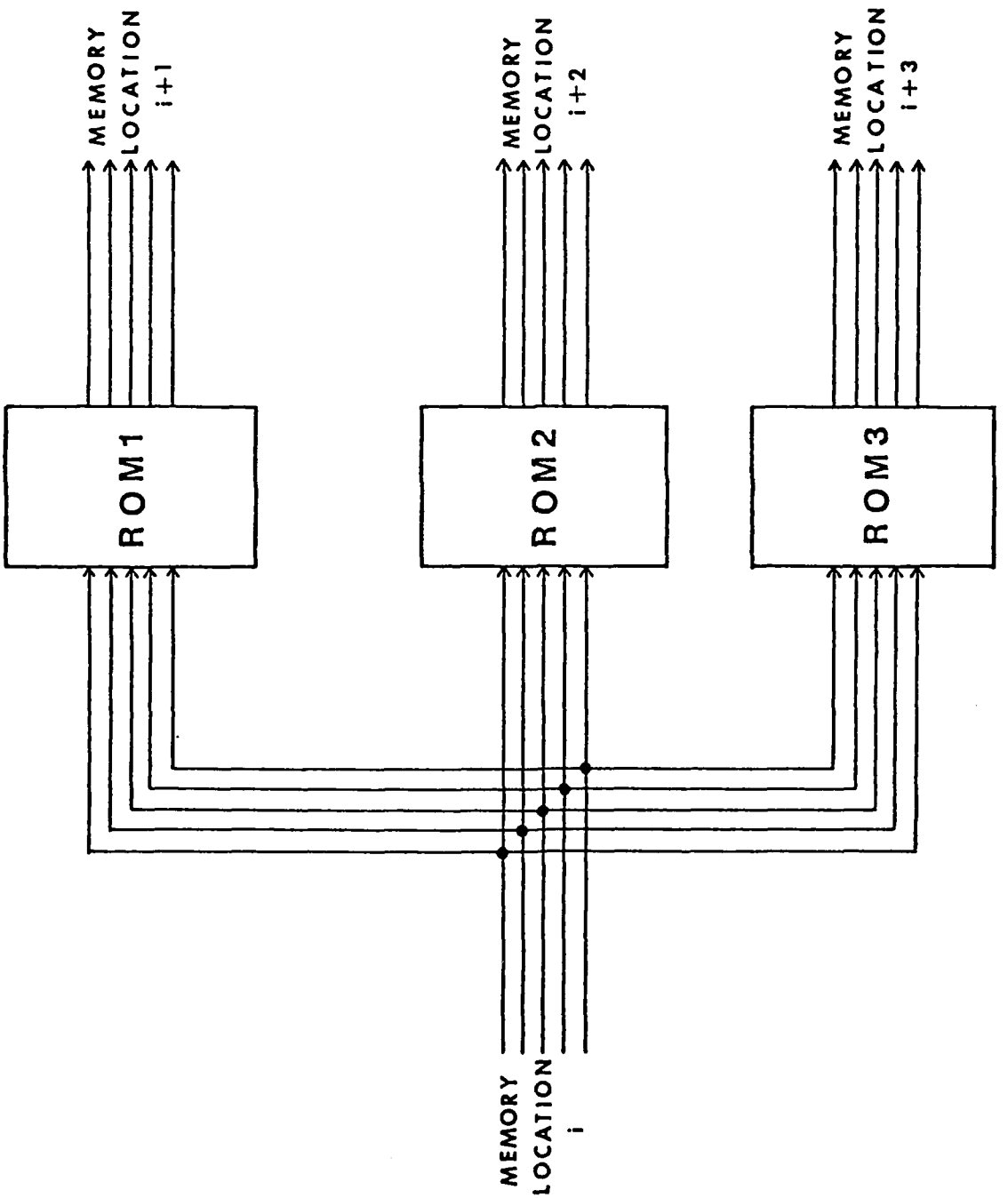


Figure 16 - Parallel ROM Look-Up

function, in order make the operation cost effective. For ordinary functions, such as SIN(X), ROM chips will be available. New architectures will probably include such arrangement; but more importantly, retrofitting existing computer should be a simple task.

Content Addressable Memory.

A content addressable memory (CAM) operates directly backwards to ROM. A data value is passed in, and the address containing that value is returned. In order to illustrate the use of such a memory, consider a ROM which produces SIN(X). If the ROM is constructed such that data can flow in, and an address will flow out; then the CAM version of the SIN(X) ROM would become an ARCSIN(X) CAM. In more general terms a combined CAM/ROM can compute both $f(x)$ and $f^{-1}(x)$. Fig. 17 shows a schematic of a CAM chip.

Semi-Custom Chips

Ultimately, the numerical analyst desires to use custom tailored circuitry to handle all needs. Unfortunately, design costs prohibit the use of custom chips. Chip makers are aware of this cost limitation and have developed components which are sufficiently specific to increase processing speed, yet general enough to cover a range of applications. These semi-custom chips come in three

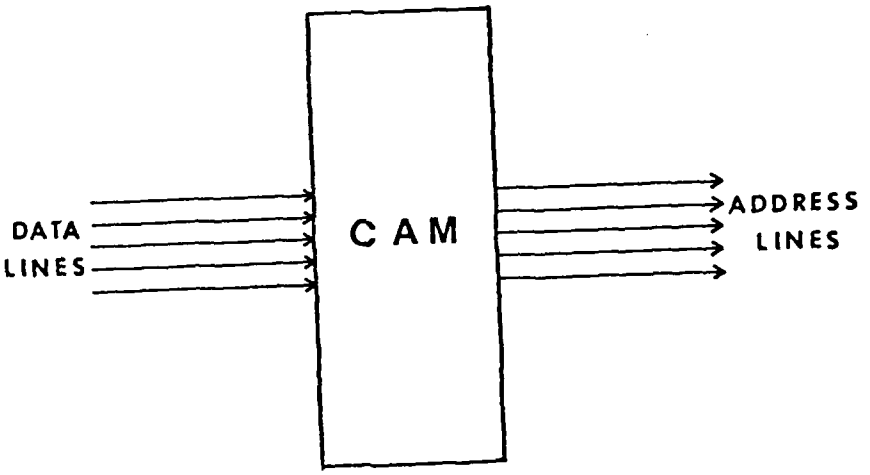


Figure 17 - Content Addressable Memory

flavors: various versions of user programmable read-only memories, processor chips with user definable instruction sets, and programmable logic arrays where the user may specify the logical interconnections of data lines.

Programmable Read-Only Memories.

Programmable read-only memories function exactly like ROM circuits (see "Read-Only Memory" section) when in operation. Unlike ROM circuits, the user defines the contents of each location once when the chip is new. After the chip has been programmed, it is incorporated into normal ROM application.

Various internal mechanism are used to implement PROM chips. Fig. 18 illustrates one of the simplest, and is presented to aid in understanding the limitation (i.e., irreversibility) of PROM circuits. Each data line in each memory location is connect to the supply voltage on one end, to the external data lines and addressing switches on the other and contains a segment of reduced diameter "wire". This reduced segment acts as a fuse element. Under normal working conditions, the supply voltage is low enough that the induced current through the fuse section of the data line does not burn out. In order to program a new chip (one where all fuses are intact), the supply voltage is raised. When any data line is connected to ground, sufficient current flows through the data lines to burn out the fuse.

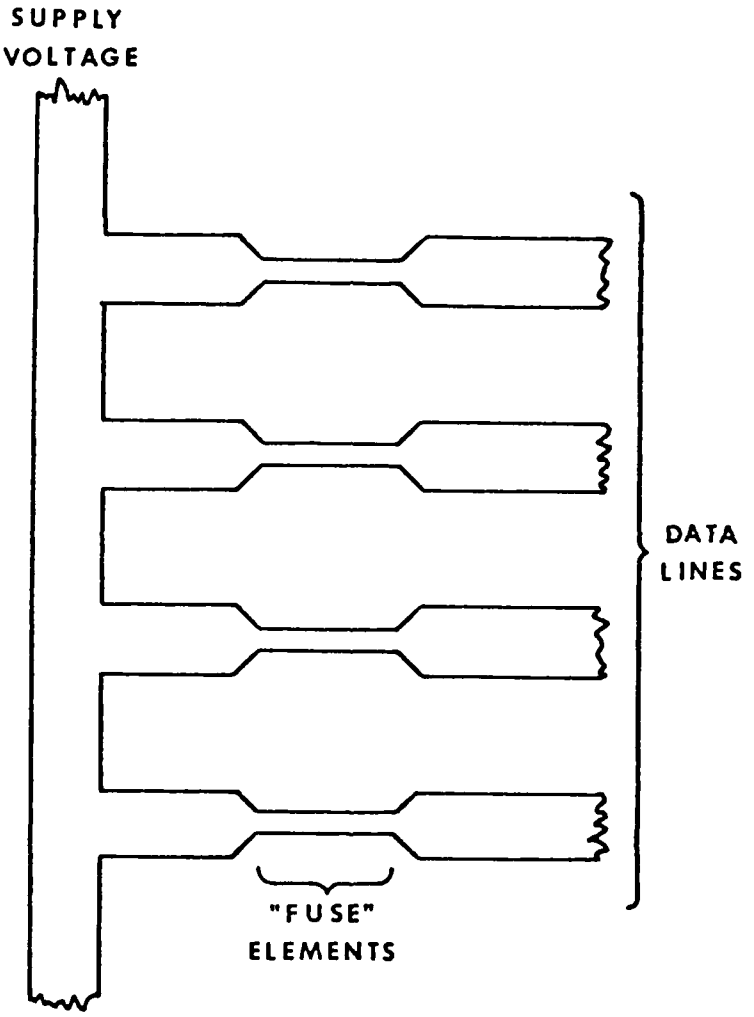


Figure 18 - PROM Circuit

If a zero (no voltage) bit is desired, the fuse is burned out; and vice versa. It should now be clear that once a zero bit is "burned in" to a ROM, it cannot be changed.

Erasable programmable read-only memory circuits use a different fusing mechanism, one that can be repaired. Repairs are usually done on a global basis, essentially creating a new chip that must be completely re-programmed. A common arrangement uses ultaviolet light to refuse the junctions.

Micro Programmable Chips.

A processing chip custom design for an application is the fastest method of computation, provided the chip is properly designed and manufactured. Two factors undermine such a technique for the numerical analyst, however: high costs for small quantity production, and ignorance of VLSI circuit designs combined with inaccessibility to the tools of the trade. Chip makers, realizing this desire, have developed micro-programmable chips to eliminate the manufacturing steps. The increased generality was traded for some efficiency.

A micro-programmable processor, also called an unbound processor, is a general purpose processing device whose specific tasks are determined after manufacture through micro-code programming. Microcode is a very primative machine language which allows the user to define the flow

and combination of data that will occur in response to an external (off-chip) operation code. Because the user specifies the responses to the various op-codes (operation codes) a custom processor can be created.

Microcode programming is more difficult because of primative programming constructs and because the programmer must keep track of timing difficulties. More advanced microcode programs provide constructs such as stacks, subroutines and DO loops.

How then, does microcode programming have an advantage over ordinary SISD processor organizations? First by allowing the user to create any arbitrary instruction set. Then, by issuing one high level command, an entire custom sequence (e.g., vector inner product) will be executed. This facilitates the creation of custom languages. Second, by using computation circuits for both data and control signals, and thus reducing the amount of needed circuitry. This technieques is a form of multiplexing. Third, microcode programs can be directly converted to programmable logic arrays (to be discussed shortly). This direct conversion simplifies hardware implementation because the algorithm can be verified, and the hardware version will have increased computation speed.

The fundamental question for implemenation by the numerical analyst is what elementary operations should be

written in microcode.

Programmable Logic Arrays.

Because all data in a computer is binary encoded, all operations (e.g., addition) are done by combining, recombining, and moving bits (on/off states). For any two binary signals, there are a total of sixteen possible ways to define the combination of the two signals. All of the combinations can be generated using one or more Boolean logic functions[24]: NOT, AND, OR, NAND, NOR, XAND, and XOR.

Custom processing devices, then, are simply circuits which combine and transfer bit in a specific pattern. And, further, all custom processor chips can be composed of series of Boolean logic circuits. Fig. 19 illustrates a simple addition circuit (no carry into the computation). It is constructed of only AND, OR, and NOT logic functions which are extremely easy to implement in VLSI technology. In order to handle complete data items, several series of Boolean logic circuits are placed in parallel.

Programmable logic arrays are simply integrated circuits which provide parallel series of Boolean logic circuits. Fig. 20 illustrates a simple programmable logic array. A binary encoded data item enters on the lower left. Desired bits are AND combined and flow out to the right by creating junctions where the data and intermediate lines intersect (junctions are shown as dots). This first set of

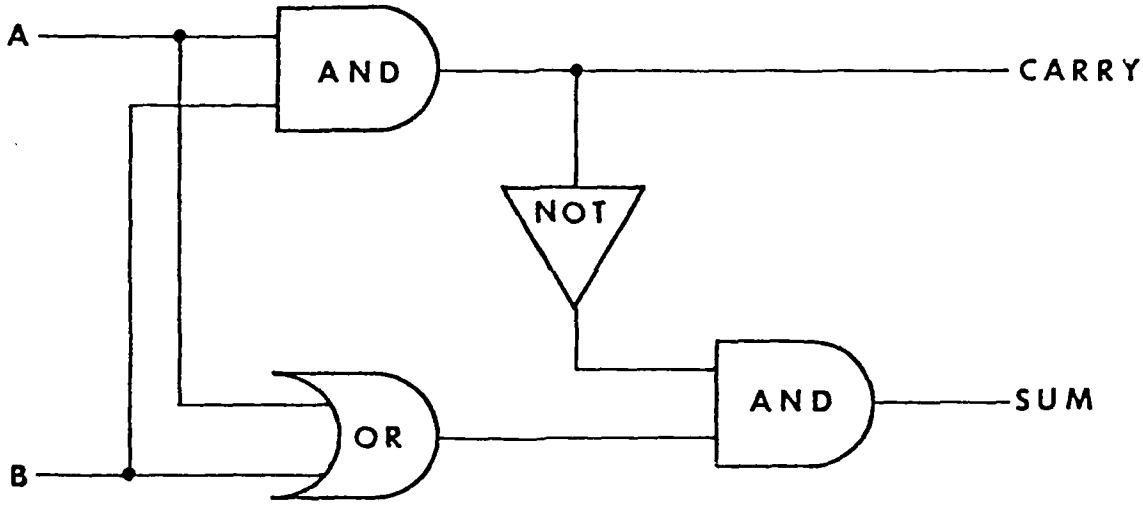


Figure 19 - Simple Adder Circuit

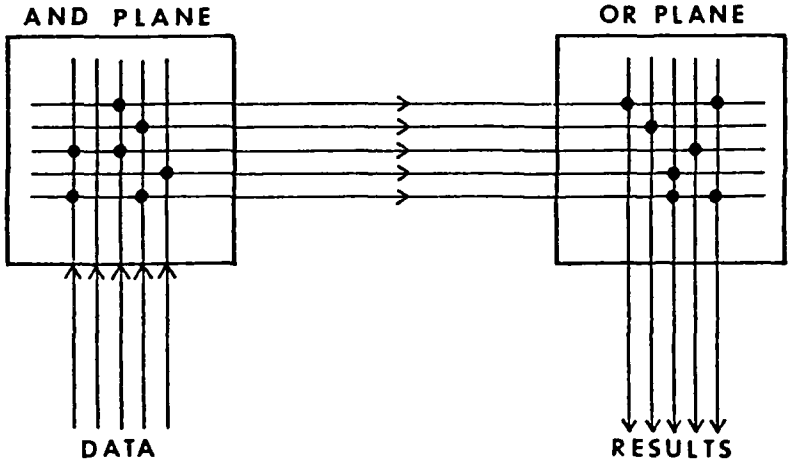


Figure 20 - Programmable Logic Array

combinations occurs in the "AND plane". The intermediate results enter the second plane from the left, desired bits are OR combined by create transistors at appropriate intersections. The final result flow out at the lower right. The second set of combinations occured in an "OR plane".

Any number of logic planes may be linked together to obtain any combination of bits. The resulting circuit is a pipeline processor composed of simple Boolean logic functions and cross-connections. The junctions can be explicitly programmed during design (or layout) of the chips, or can be made user programmable using techniques similar to those in PROM circuits.

The user programmable version of programmable logic arrays offer the numerical analyst an inexpensive method of creating custom processing devices. The primary problem is, however, which bits should be combined and with which Boolean function.

Custom VLSI Chips

Chips custom designed for a particular application are, if properly designed, the fastest computing tool available. Unfortunately, design costs for custom chips are very high and manufacturing cost for small production runs are also

expensive. These two factors generally prohibit the used of custom VLSI circuits by the numerical analyst for structural engineering computations. Both design and manufacturing costs are dropping rapidly, and a brief overview is in order for when (not if) it becomes economically available. Further, some special processor chips are emerging which are directly applicable. Both types of review follow.

Gate Arrays.

Gate arrays [8][23] are the hardware version of programmable logic arrays. They contain arrays of Boolean logic functions, but the connections are manufactured rather than programmed. Also, because the junctions are specifically designed, junctions which will not be needed are not created. In this fashion, the number of transistors and chip area is reduced. This allows shorter communication paths and high processing speeds.

Hartmann[8] has compiled information on commercial availability of gate arrays and predicts densities of 10,000 gates/chip by 1985. Presently, chip capacities vary from 50 to 2,000 gates which is sufficient to be useful in special purpose structural engineering computations.

Design Tools.

The design of a VLSI chip is said to be equivalent to the design of a traffic system for New York City. The tools required to handle such a problem are either strategic in

nature or are computer aided design procedures. Because of the scope of CAD programs, they are considered in a separate section.

The primary dictate in VLSI designs is: control complexity through modularity. This is aided by maintaining regular data and control flow. The polycell[13] technique is a classic implementation of modular design. Each polycell is composed of a library of standard function blocks; thus, rather than having to design a desired circuit, the designer simply connects to the appropriate input and output terminals. The polycell, once designed, becomes a black box. The design of the polycell itself is still a major obstacle, but as time progesses many "off the shelf" models will be available. Of more concern, is the inherent waste of the polycell technique; usually only one of the many possible functions is used and the circuitry for other functions is wasted. In some cases, several of the polycell connections can utilized, but it is rare that all would be used. The excess circuitry, forces longer communcation lines and, thus, longer processing times. The trade-off is between processing speed and design time.

High manufacturing costs often prohibit the development of small or experimental VLSI chip circuits for custom applications. Universities teaching VLSI design were keenly effected, and jointly implemented a scheme called the

multiproject chip.[6] Several integrated circuit designs were placed on the same set of masks, each complete and independent processing devices. A single chip (multiple copies, however) was then fabricated and returned to the designers who made their own off-chip connections. Because the manufacturing costs had been distributed between several users, the average cost per project amounted to less than $500 (manufacturing costs only). This route appears promising for small scale custom integrated circuit implementation and experimental work.

The multiproject procedure also served to illustrate the needs for standards and communication between designers and manufacturers. Careful planning and adherance to the master plan was critical to making the multiproject chip scheme operate. In addition, many CAD tools were utilized; each project was completed within three month time. A total of 82 projects, created by 124 designers, were implemented.

CAD Programs.

Integrated circuits cannot be designed without a computer (utilizing conventional techniques). The mask is almost always drawn by a computer, and virtually must be drawn by computer when implementing VLSI designs because of the volume of features to be included. Fig. 21[12] illustrates the complexity of a medium scale integrate circuit.

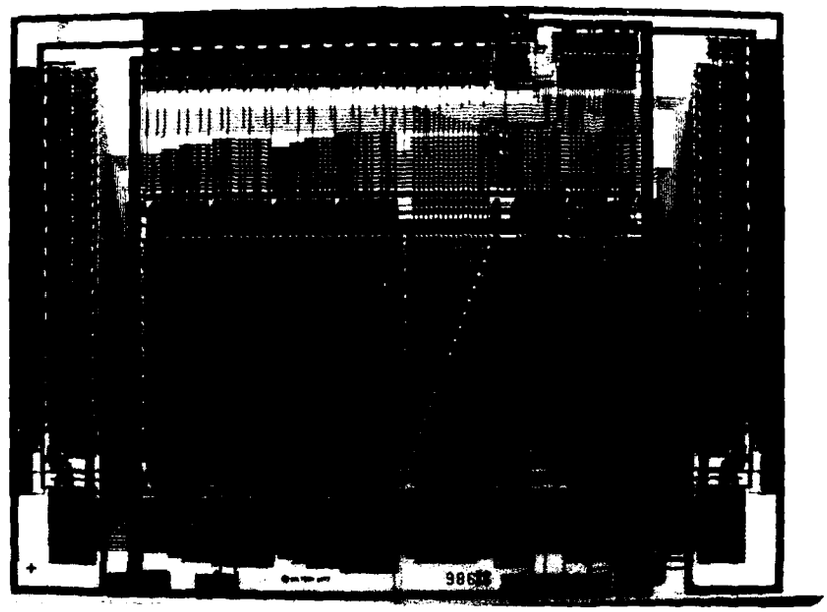


Figure 21 - OM2 Chip

Programs which are used in the manufacturing process are essential programs. They consist mostly of pattern generators and numerical control (computer driven machines) programs. They accept "design files" as input. In order to process the design files, a standard is required. Originally, each manufacturer had his own requirements for the design file; but presently, all manufacturers handle design files which meet the California Institute of Technology Intermediate Form, also called: CALTECH Intermediate Form and CIF. An overview is presented by Mead and Conway [20]. In addition to input programs, some type of output is essential to verify the input. This output is generally presented in a "check plot". The check plot is a magnified version of what will actually be placed upon the chip.

All integrated circuits can be traced back to some point at which the junctions were entered into the memory of the machine by hand. This process is called "hand digitizing". It is certainly possible to hand digitize the entire design of a chip. If we assume that a single junction can be digitized in one minute, then a circuit with 480 junctions could be entered (not designed) into a computer in one day. For a VLSI design of 500,000 transistors, hand digitizing would require about 125 weeks. Adding design time, a reasonable approximation is 15

man-years. This length of time is not acceptable, and further the repetitive nature of hand digitization is tedious and would inevitably lead to mistakes.

At a minimal level, VLSI CAD programs must serve as automated drafting systems.[21] The user must be able to digitize an object once to create a basic symbol which can then be manipulated and recombined to form more complicated symbols. On Digital Equipment Corporation's floating point processor only 1000 of 77,000 transistors were hand digitized. This is more than a 98% savings over complete hand digitization. Programs of this type are simply aimed at increasing productivity within the conventional design process.

Discovering that a chip will not function due to design error or oversight after manufacturing is both disappointing and wasteful. One important class of CAD tools are those which test the validity of a design from design files. This class of tools check the design file against design rules[25], electrical rules, and simulate[5] the response of the final chip. Baker and Terman[3] describe one such tool. The use of these tools adds two to three days to overall design time (based upon a three month design), but the increase in probability of a function chip is well worth the time. Of course, such checking programs utilize the non-dimensional constant lambda, and are thus, applicable to

all implementation technologies.

The final class of CAD programs allow the user to describe the design at a higher level, and "compile" the specifications into a geometric and topological configuration. This higher level allow the creation of macros, a sort of subprocedure which allow the creation of specialized extentions to the high level specifications. Bristol Blocks,[22] created by David Johanssen of Cal. Tech., is a program which provides high level specification of a VLSI design. While limited to design of one type of microprocessor, the power of the program is obvious. The use specifies, in two or three pages, what each block of the microprocessor is to do. The program designs the blocks, determines where connections must be made, arranges the blocks, and makes the connections. The final output is compatible with manufacturing programs. Preparing the input requires some effort, but the program reduces design time from 3 years to a few minutes.

Conclusions

The numerical analyst is not versed in VLSI design; the VLSI designer is not versed in algorithmic development. A primary implication of VLSI technology on the numerical analyst is locating the communication interface in order to

optimize the talent of the numerical analyst and the VLSI designer. A viable alternative is to automate the algorithmic translation, accepting the inherent inefficiencies, and allow the numerical analyst full control of VLSI implementation.

Regardless of the mechanism for producing the final integrated circuit design, the present approach to algorithm development must change to include concurrency in programming and regular flow of data and control. Specifically, the developer must overcome the ingrained notion that processor are expensive; interconnection paths are more expensive in time and costs than junctions.

Acknowledgement

The funding for this research was provided by the Office of Naval Research under contract no. N00014-80-C-0712.

References

1. Anon.,"VLSI Design Tool Development at DEC", Lambda, Redwood Systems Group, Palo Alto, California, Vol. 2, No. 1, 1981, pp. 12-13.

2. Anon., Home Computers, Publications International, Ltd., Skokie, Illinois, 1978, pp 121-122.

3. Baker, Clark M., and Terman, Chris, "Tools for Verifying Integrated Circuit Designs", Lambda, Redwood Systems Group, Palo Alto, California, Vol. 1, No. 3, 1980, pp. 22-30.

4. Braun, Jerry, and White, George, "Parallel Processing with Minicomputers Increases Performance, Availability", Electronics Magazine, July 5, 1979, pp. 125-129.

5. Bryant, Randal E., "An Algorithm for MOS Logic Simulation", Lambda, Redwood Systems Group, Palo Alto, California, 1980, Vol. 1, No. 3, pp. 46-53.

6. Conway, Lynn, Bell, Alan, and Newell, Martin, "MPC79: The Large-Scale Demonstration of a New Way to Create Systems in Silicon", Lambda, Redwood Systems Group, Palo Alto, California, Vol. 1, No. 2, 1980, pp. 10-19.

7. Gallagher, R. H., "Computerized Structural Analysis and Design - The Next Twenty Years", Second National Symposium on Computerized Structural Analysis and Design, George Washington University, 1976.

8. Hartmann, Robert F., "Design and Market Potential for Gate Arrays", Lambda, Redwood Systems Group, Palo Alto, Vol. 1, No. 3, 1980.

9. Lyon, Richard F., "Simplified Design Rules for VLSI

Layouts", Lambda, Redwood Systems Group, Palo Alto, California, 1981, Vol. 2, No. 1, pp. 54-59.

10. Mead, Carver, and Conway, Lynn, Introduction to VLSI Systems, Addison-Wesley Publishing Company, Reading, Massachusetts, 1980, pp. 266.
11. Op. Cit. 10, pp. 270.
12. Op. Cit. 10, pp. Front Cover.
13. Op. Cit. 10, pp. 62.
14. Op. Cit. 10, pp. 195.
15. Op. Cit. 10, pp. 295.
16. Op. Cit. 10, pp. 360.
17. Op. Cit. 10, pp. 52.
18. Op. Cit. 10, pp. 35.
19. Op. Cit. 10, pp. 45-46.
20. Op. Cit. 10, pp. 115-127.
21. Newman, William, and Sproull, Robert, Principle of Interactive Computer Graphics, McGraw-Hill Book Company, New York, 1979.
22. Robertson, Arthur L., "Are VLSI Circuits Too Hard to Design", Technology Review, MIT Press, Cambridge, Massachusetts, January 1981, pp. 52-53.
23. Roffelsen, Larry, and Jansen, William D., "Gate Arrays: A User Perspective", Lambda Redwood Systems Group, Palo Alto, California, 1981, Vol. 2, No. 1, pp 32-36.
24. Rooney, Martin F., "Computer Hardware for Civil

Engineers", Journal of the Technical Councils, American Society of Civil Engineers, New York, 1981, Vol. 107, No. TC1, pp. 153-168.

25. Whitney, Telle, "A Hierarchical Design-Rule Checking Algorithm", Lambda, Redwood Systems Group, Palo Alto, California, 1981, Vol. 2, No. 1, pp. 40-43.

Appendix - Design of VLSI Circuits

The numerical analyst will not be called upon to actually design a VLSI chip, but a basic understanding of the rudimentary concepts and limitations involved in such a design is necessary to communicate desired performance and utilize special capabilities of VLSI chips. With this goal of establishing minimal conversency, this chaper will present brief discussions on type of VLSI technologies, color stick diagrams (the medium of communicating VLSI designs), fabrication procedures, physical limitations, and yield theory (a method of predicting manufacturing success).

Choice of Technology.

There are three workable technologies that can be used to implement VLSI chips. The first is nMOS or n-channel metal oxide silicon. The second is CMOS or complementary metal oxide silicon. And the third is I^2L or integrated injection logic. There are several factors which will affect the choice: circuit density, unit power performance, richness of circuit functions (i.e., number of elementary circuit components available), topological properties of interconnection paths, suitability for implementing the total system on

one chip, and availability of manufacturing facilities. The numerical analyst should not be concerned with the differences as that decision is made by the circuit design. However, the numerical analyst should be aware that implementation requires more than simply transcribing the algorithm to circuitry, and further, should not be unnerved by such acronyms. To the numerical analyst, the result implemented with any technology will appear the same.

Fabrication.

MOS technology integrated circuits are made of three layers of conducting material separated by layers of insulation.[1] The bottom conducting layer is called the diffusion layer; next is the polysilicon layer; and a metal layer is on top. No layer is continuous over the chip surface, but has a pattern. In addition, cuts are made in the insulation at select points to make connections between the layers. The key to the technology is that a transistor (or junction) is formed where ever the polysilicon layer passes over the diffusion layer. Fig. 22 illustrates the physical occurence and corresponding electronic symbol. This transistor is used in digital circuits as a simple

---

[1]The majority of this section was adapted from Introduction to VLSI Systems, by Carver Mead and Lynn Conway, Addison-Wesley Publishing Company, Reading, Massachusetts, 1980.

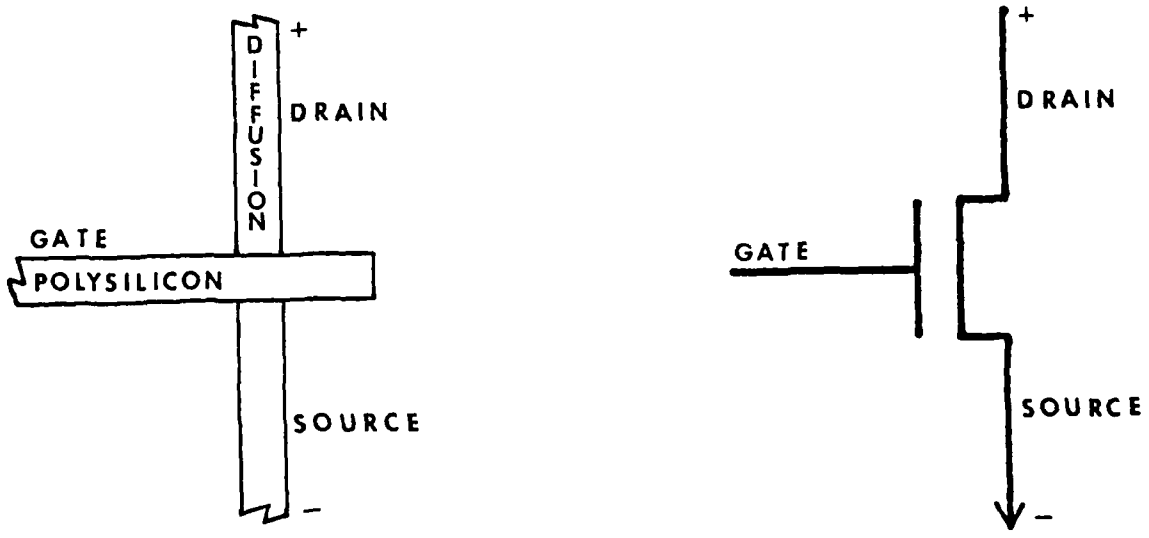


Figure 22 - MOS Transistor

switch: when voltage is applied to the polysilicon gate, current flows between source and drain on the diffusion layer. No voltage on polysilicon results in no flow in the diffusion layer. Thus, this junction behaves as a normally open (turned off) switch, and is designated as an enhancement mode MOSFET (Metal Oxide Silicon Field Effect Transistor).

The patterns for each layer are placed on the chip using a process similar to ordinary photography. Several distinct steps are required:

1. Form Mask. A mask is a transparent material with opaque areas. Masks may be positive or negative, like photographic slides and negatives. An exact duplicate of the mask is made, so the mask must be same size as the final chip. (Further discussion on mask making will be included under "CAD Programs".)
2. Treat Surface. The surface of the chip is treated in two steps. First, a layer of silicon dioxide is formed by baking the bare silicon wafer. Second, a layer of resist is baked onto the layer of silicon dioxide. This resist is a photo-sensitive organic compound which will be used to capture the mask image and resist the effects of certain chemicals.
3. Exposing the Chip. The mask is positioned over the treated wafer, aligning the mask with any previous

masks. The wafer is then exposed to radiation (e.g., ultraviolet light) which passes through the mask. The opaque areas block the radiation, and the pattern is transfered to the resist.

4. Develop Resist. The sensitized resist is now developed like photographic film. If a positive resist was used, the exposed areas are disolved; if a negative resist was used, the unexposed areas are disolved.
5. Etch the Wafer. The exposed silicon dioxide (i.e., the areas under the disolved resist) are etched away with a chemical, usually an acid. The resist is not effected by the etching chemical, and protects the silicon dioxide below it.
6. Remove Resist. The remaining undisolved resist is now disolved to leave a patterned layer of silicon dioxide.

The above steps are shown in Figs. 23 and 24. The process must be repeated for each layer.

Besides the three conducting layers of the chip, two additional processes will require masks. The first is the implant areas. Transistors are normally open, but by implanting ions into the silicon surface, transistors can be made normally closed. The areas not to be implanted must be protected with resist, so a mask is used. Normally closed (turned on) transistors are called depletion mode MOSFET's, and the implanted area is the implant layer. This

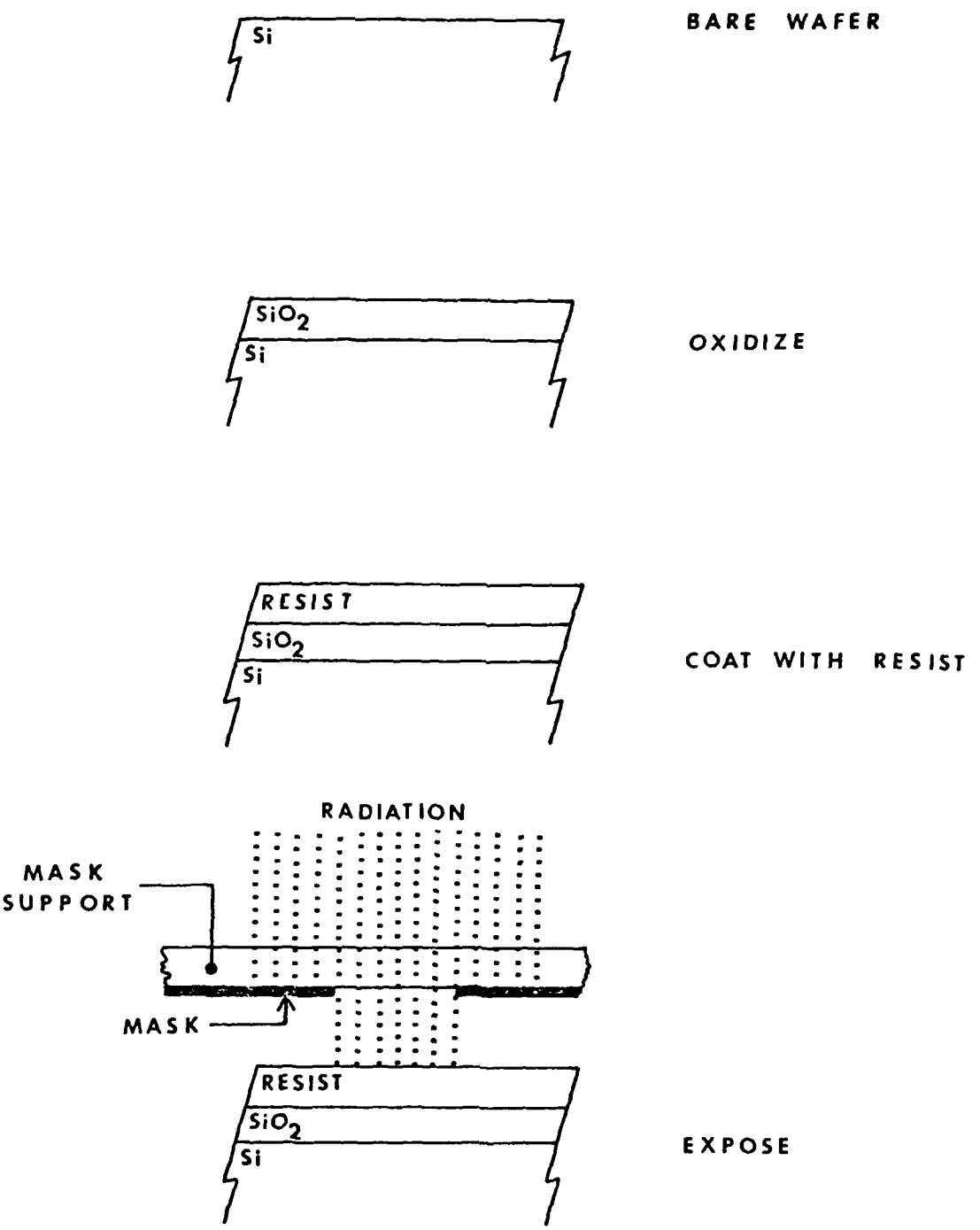


Figure 23 - VLSI Fabrication

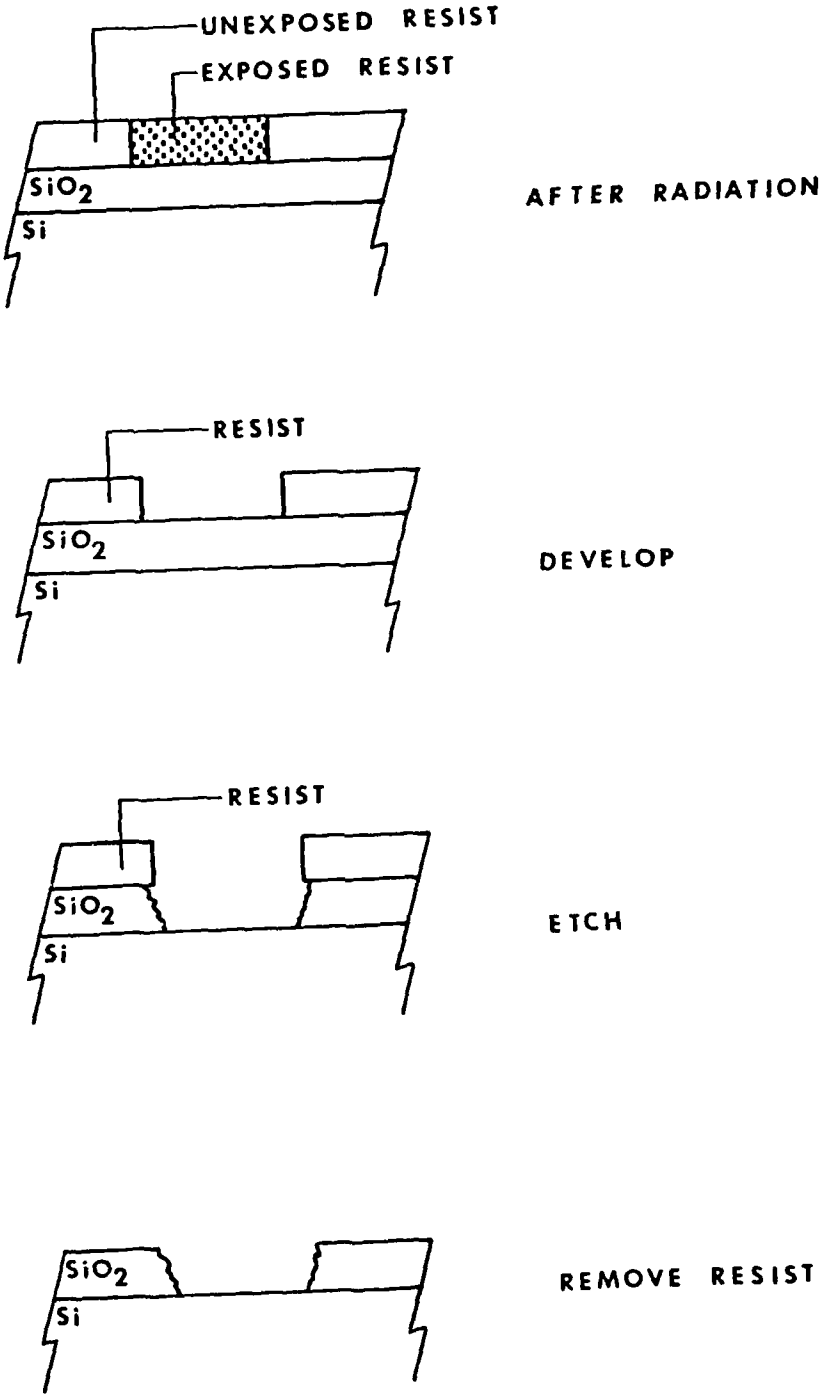


Figure 24 - VLSI Fabrication

implanting is done between the diffusion and polysilicon layers, and does not actually form an additional layer. The second addition mask is used for overglassing. Overglass is used to protect the chip, but must have some openings to allow leads to be connected.

Stick Diagrams.

Color stick diagrams may not be of great concern to the numerical analysts seeking to use VLSI technology, but minimal conversance with the technique may prove useful in understanding technology advances and design difficulties. More importantly, however, color stick diagrams are the medium used to communicate VLSI design information. As with any discipline, a common communications link is essential.

Integrated circuits are three dimensional, that is, composed of distinct layers with specific properties. Integrated circuit design is done on a two dimensional medium, paper; and a need to express the three dimensionality is needed. Color is used to illustrate depth, or specifically, the layering of the chip. In addition, certain layers may have their properties changed through chemical and/or electromagnetic radiation procedures, and color is also used to illustrate effected areas.

A quasi-standard has emerged for color indication of each layer:

1. White - Standard background color, indicating bare silicon wafer with no layers above.
2. Green - Diffusion layer, this is the first layer on the chip. This layer contains sources, and drains for all transistors. It may also be used for interconnection paths.
3. Yellow - Depletion mode designation. These markings are used to indicate area that will be specially treated to change the material properties of the layer below. If left untreated, transistors are normally open (turned off). Treated areas, marked in yellow, create transistors that are normally closed (turned on). The yellow markings are not actually a layer of the transistor.
4. Red - Polysilicon layer (often abbreviated as the poly). This layer forms the gate of the transistor, and is also used for interconnection paths. At all locations where the polysilicon layer passes over the diffusion layer (a red line intersects a green line), a transistor is formed. If the area is marked in yellow, a normally on transistor is formed, called a depletion mode transistor. If the area is not marked in yellow, a normally off transistor is formed, called an enhancement transistor.
5. Blue - Metal layer. This layer indicates the location

of metal paths in the chip. Metal paths can be used for any communication purpose, but generally carry supply voltages and ground connections. Digital Equipment Corporation now uses two metal layers on some chips[1], and it is not known how these similar layers are distinguished.

6. Black - Interlayer connections. Black area are used to indicate connections between layers which are normally isolated by intermediate layers of insulating material.

The color selected to designate each layer is arbitrary, and has no bearing upon the transistor itself. The colors are used to aid in human recognition.

To illustrate the simplicity of the technology, a NAND gate circuit is shown. This particular circuit has the property that it produces a "1" (or positive voltage) unless both input A and B are "1". The schematic diagram appears in Fig. 25. If both A and B are on, then current will flow through their drains and sources and pull the voltage of the output to ground (zero volts). If either A or B are off, then no current will flow, the output will be isolated from ground and will rise to the supply voltage. Fig. 26 shows the stick diagram used to communicate the VLSI version of the circuit. The input lines enter on the polysilicon level, and the output line exits on the diffusion level. The resistor is created using a depletion mode (normally on)

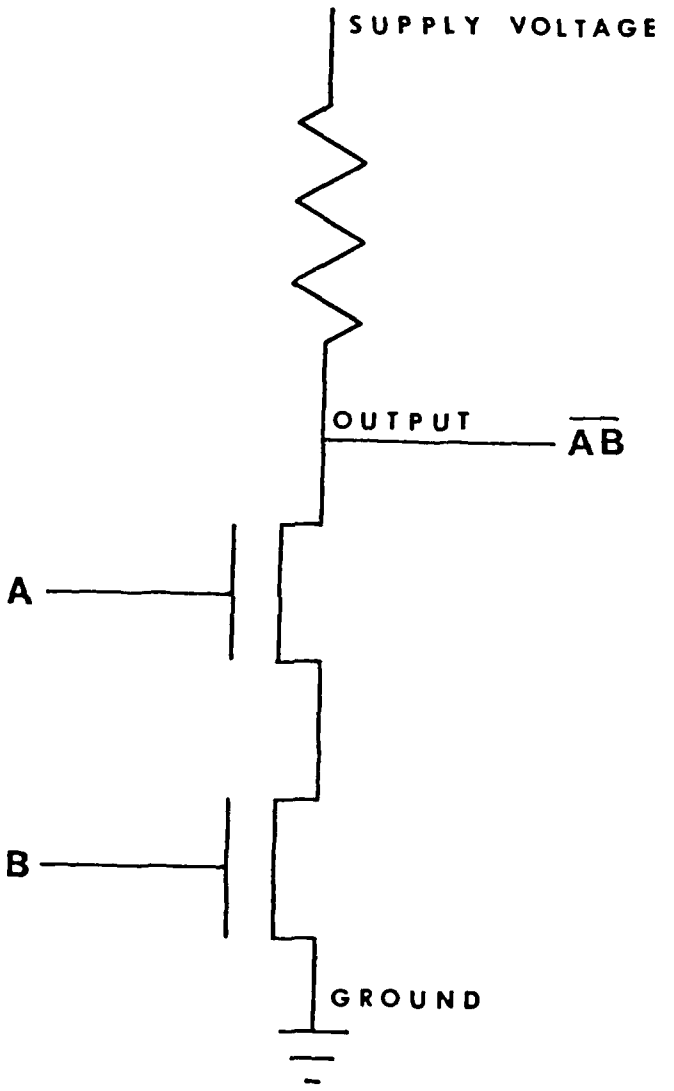


Figure 25 - Schematic NAND Gate

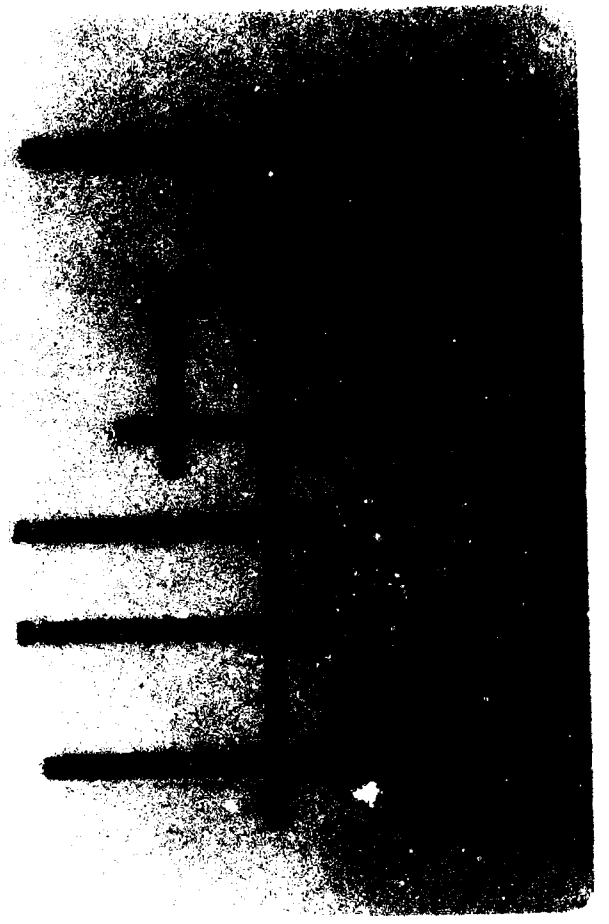


Figure 26 - Integrate Circuit NAND Gate

connected to the output line. When A and B are closed and current is flowing through them, there exists a voltage drop between supply and output which in turn causes the gate of the depletion mode transistor to turn the junction off: infinite resistance exists. When A and B are not conducting, no voltage drop exists between supply and output, and the gate turns the junction on: zero resistance. Because the intermediate resistance is unimportant, the depletion mode transistor behaves as desired.

Physical Limitations.

The trend in VLSI chip technology is to make it smaller and put more into one package. In theory, this direction should continue indefinitely, but the fundamental laws of physics along with manufacturing tolerance place limits on the extent of this downward scaling. The numerical analyst using VLSI technology would like to know the absolute limit of downward scaling. A single answer would be misleading, rather the VLSI user must know the types of limitations because some are design trade-offs, not absolute. This section will highlight some of these limitations.

Integrated circuit transistors, while very low in power consumption, do produce heat. Because heat and switching energy are both energies, there exists a probability that heat may build up and change the logic state. For thermal instability:

1. Failure can be reduced, but never eliminated.
2. As response time decreases (more changes per unit time), the failure rate inversely rises.
3. The signal energy to heat ratio must remain large. This is a dilema because the increased switching energy disipates more heat.

However, in today's technologies, thermal failure is not a problem.

Integrated circuits are tuned oscillators which are always on. As such, they are subject to electromagnetic radiation, particularly electrical noise. In order to function, the signal level must be in excess of the noise peak levels. While much can and is done to reduce such noise, it cannot be eliminated. Thus, a minimum signal level is set, but varies with the operating environment of the chip. This limitation is many times that for thermal failures.

Tunneling is a phenomenon caused by the wave nature of the electron. When an electron encounters an electrical barrier, it is reflected back. The reflection is not abrupt and motion decays exponentially. If the barrier is too small to allow sufficient distance to reverse the direction, the electron will pass the barrier. As electrons possess various quantum amounts of energy, a percentage alway pass the barrier; the quantity which do are collectively called

leakage. For one eV, the critical boundary (one where all electrons pass) is .001 microns. 1978 technology had already used gate oxides of .1 microns. This quantum factor is, thus, a rapidly approaching absolute limit.

Superconductors, circuits running at temperatures close to absolute zero, have received considerable attention. They are expected to reach $5*10^{12}$ instructions/second in the near future, while FET (field effect transistors) are only expected to read $5*10^{10}$ instructions/second. Despite the two orders of magnitude improvement, they are inherently inefficient for terrestrial applications where ambient temperatures are far above absolute zero. In order to remove the heat, work must be done. Circuit designers [16] presently feel that the economics of providing such cooling power make the use of superconductor economically unfeasible in conventional applications. Further, superconductors have special problems in communicating with non-superconductor elements which must be used to interface with the outside world. Superconductors are, therefore, limited by economics and communications.

The metal layer in integrated circuits is subject to a special problem known as metal migration[17]. When a critical current density is exceeded in a metal conductor, metal atoms are pulled along with the current. Current density is inversely related to cross sectional area of the

metal lines, and the migration reduces cross sectional area and further aggrevates the situation. If allowed to continuously occur, open circuits in the metal lines result and render the chip useless.

From the preceding paragraphs it is obvious that the technology is approaching limits imposed by physics. Carver and Mead [18] present a concise table for physical limitation:

Item	1978	Ultimate
Minimum Feature Size	6 microns	0.3 microns
Junction Delay Time	0.3 nsec	0.02 nsec
Switching Energy	10^{-12} joules	10^{-16} joules
System Clock Period	30 nsec	2 nsec

Beyond the limitations imposed by physics, the manufacturing process has some practical limitations. One of the primary manufacturing limitations is mask alignment. Runout is an alignment problem which occurs due to successive mask alignment. Because each pattern is aligned with the immediately preceding mask, significant error can accumulate. This type of alignment is emphasized as feature sized decreases (circuit density increases). Wafer distortion is the other major source of alignment difficulties. Because high temperatures are used in baking on the silicon dioxide and resist coatings, the wafer

distorts. Due to thermal straining, the wafer does not return to its original shape upon cooling. While the masks are aligned perfectly, the boundaries on the chip have changed. As chip size increases these effects become more prominent. the chip.

All chip designs must conform to certain design rules[9], in order to guarantee electrical performance. These rules specify minimum line widths of the communication lines on each layer, the separation between lines, the amount of overlap in a junction, and the amount which a line must extend past another to assure contact. Further, the values change for each type of technology. The checking for conformance to the rules grows exponentially with the number of junctions on the chip, and is virtually imposible to accomplish by hand. Fortunately, the rules may be specified in terms of a non-dimensional unit designated as λ (lambda). λ is solely a function of the technology selected, and serves to isolate design decisions from manufacturing constraints. CAD programs (to be discussed in "CAD Programs") are used to combat the combinatorial nature of rule checking. The numerical analyst should be keenly aware of the substantial time and man-power requirements need to verify conformance.

In VLSI technologies, the cost of interconnection wires is more than the junctions in both time delays and dollar

expense. The scarce resource in VLSI chips is, therefore, communication bandwidth. The planning for regular flow of data and control information is a major difficulty because most algorithms developed to date do not consider information flow, but rather are based upon minimizing processor requirements. A related problem is ground routing. Existing electronic circuit schematics (the circuit equivalent of an algorithm) assume ground to be ever present, yet in a chip this ground path must be explicitly designed. Ground routing and supply voltage routing further complicate the communication bandwidth difficulty by requiring two more communication signals beyond data and control lines.

Yield Theory.

All previous chip discussions implicitly assumed the wafer to be perfect, but this is not the case. The downward scaling has increased the demand for perfect silicon wafers because even small flaws are of the same magnitude as feature size. Flaws that cause a chip to malfunction are known as fatal flaws. Because flaws are extremely small, it is not feasible to scan and detect them; and further, many do not occur until the manufacturing steps are well underway.

Understanding that such flaws exist and cannot be detected, a statistical procedure is used to determine the

probability that a specific chip will function when made. These statistical procedures are called yield theory[19].

Related to yield theory is testing. Two goals are desired from the test results: is the entire chip working correctly; and if not totally correct, is it salvagable. Testing of junctions and their interconnections is not a trivial task, however. Like design rule checking, testing of chips grows exponentially. The complete testing of most VLSI chips is not remotely feasible, but improvements in the extent of testing are being made. The present general solution, is to test certain key circuits of the VLSI chip, and assume the remainder are correct. Further, most chips contain some standard test circuit which is used to assure that the manufacturing steps were completed correctly.

Glossary

λ - Lambda is the fundamental unit of length in integrated circuit design. When an appropriate value is substituted for a given technology, actual dimensions of features on the chip may be obtained.

ALU - Arithmetic Logic Unit, the processing element of a computer or chip.

Array Processor - A special purpose peripheral device which performs matrix operations with a single command.

Bandwidth - A measure of the capacity of a circuit to communicate, equal to the number of "wires" or communication lines connecting two devices.

Bus - Collective name for a group of wires carrying one type of information (e.g., a data bus).

Cache - A communications technique, usually associated with memory which stores data items from a single communication line, then passes them along many data lines. Also operates in reverse fashion.

CAD - Computer Aided Design

CAM - Content Addressable Memory (also used in other disciplines as Computer Aided Manufacturing)

CIF - CALTECH Intermediate Form, a standard used to transfer

the geometric and topological design of a chip between designer and manufacturer.

CMOS - Complementary Metal Oxide Silicon, one of the technologies for implementing VSLI Circuits.

Color Stick Diagrams - A graphical method for conveying an integrated circuit design. Each color represents a layer of the chip: green=diffusion, red=polysilicon, blue=metal, white=bare wafer, black=connections, yellow=implant areas.

Concurrency - Concept of a global processor with internal processors operating simultaneously.

Depletion Mode - A transistor that is normally on, and when charged shuts off. Shown as the junction of green and red lines on a yellow (implant) background on a color stick diagrams. Also, can be configured to operate as a resistor.

Diffusion - The bottom layer on an integerated circuit, shown as green on color stick diagrams.

Digitize - To enter the geometric and topological layout of a design or component into the computer, usually through a graphical input device.

Enhancement Mode - A transistor which is normally off, and when charged, turns on. Shown as the intersection of red and green lines on a white background on color stick diagrams.

EPROM - Erasable Programmable Read-Only Memory. Usually, must be removed from circuit to program and/or erase. Erasure often done by subjecting to ultraviolet light.

Finite State Machine - Any processing device where some or all of all the outgoing data lines (results) are used as part or all of the next instruction.

Gate Array - A collection of Boolean logic junctions which operate on a number of data lines simultaneous. Junctions are programmed during manufacture.

Hexagonal Processor - A special type of processing device which accepts several inputs and produces one output, developed by Kung.

Hung Switch - A transistor which is caught in a metastable state, and is neither on nor off. Often called a synchronization failure.

I/O - Input Output, or Input Output device.

Interlayer Connection - An area of insulation which has been removed to allow the connection of two conducting layers on an integrated circuit. Shown as black on color stick diagrams.

I^2L - Integrated Injection Logic, one of the technologies available for implementing integrated circuits.

Macro - A user definable block of code, usually machine language, that is actually copied when called (as

opposed to jumping and returning).

Mask - A transparent base with opaque areas used to transfer an integrated circuit design to the chip, much like a photographic negative.

Metal - One of the layers of an integrated circuit. Shown as blue on color stick diagrams. Usually used to conduct supply voltages and ground.

Microcode - A primative machine language used to custom program a microprocessor.

MIMD - Multiple Instruction Multiple Data, usually a pipelined parallel processor.

MISD - Multiple Instruction Single Data, a pipeline processor.

MOSFET - Metal Oxide Silicon Field Effect Transistor

Neumann, John von - The person credited with the concept of handling programs as data, and thus, combined program-data storage.

nMOS - n-channel Metal Oxide Silicon, one of the technologies available for implementing integrated circuits.

Op-Code - Operation Code, the segment of an instruction which defines the type of operation to be performed.

Overglass - A protective layer placed on top of the chip which contain openings for leads only. Is not

indicated on color stick diagrams.

Parallel Processing - A processor organization scheme where several processor are placed side by side, and all connected to the same instruction lines. Allows several data to be operated upon simultaneously.

Pipeline Processing - A processor organization scheme where several processors are concatenated by connecting the output data lines of one to the input data lines of the next. Allows one instruction to send a datum through several operations.

PLA - Programmable Logic Array, a group of Boolean logic junctions, user programmable, which operate on a number of data lines simultaneously.

Polysilicon - The second layer in an integrated circuit. Shown as red in color stick diagrams.

PROM - Programmable Read-Only Memory.

Resist - Organic compound used in manufacturing integrated circuits to resist the effect of etching chemicals. Is photo-sensitive and is processed like a photographic film to transfer the design to the silicon wafer from the mask.

ROM - Read-Only Memory.

Silicon Dioxide - The materal used to form the conducting layers of an integrated circuit.

SIMD - Single Instruction Multiple Data, a parallel

processor.

SISD - Single Instruction Single Data, a conventional processing device.

State Lines - The connecting wires between output data lines and instruction lines in a finite state machine.

Superconductor - A transistor or integrated circuit run at temperatures close to absolute zero, thus operates at extremely high rates of speed due the increase conductivity of materials at those temperatures.

VLSI - Very Large Scale Integration

Yield Theory - A statistical procedure used to predict the number of successful chips from the manufacturing process.

Index

AD-A101 777 RENSSELAER POLYTECHNIC INST TROY N Y F/G 9/2
VLSI TECHNOLOGIES AND NUMERICAL ANALYSIS,(U)
JUN 81 L J FEESER, M F ROONEY, M S SHEPHARD N00014-80-C-0712
UNCLASSIFIED NL

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A101 777	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) VLSI TECHNOLOGIES AND NUMERICAL ANALYSIS		5. TYPE OF REPORT & PERIOD COVERED Interim Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) L. J. Feeser, M. F. Rooney, M. S. Shephard		8. CONTRACT OR GRANT NUMBER(s) N0014-80-C-0712
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rensselaer Polytechnic Institute, Department of Civil Engineering, Troy, New York 12181		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Director Structural Mechanics, Material Science, Office of Naval Research, 800 No. Quincy Street, Arlington, VA 22217		12. REPORT DATE June 29, 1981
		13. NUMBER OF PAGES 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computers, Numerical Analysis, Very Large Scale Integration, Integrated Circuits, Computer Aided Design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This technical report explains the hardware alternative of Very Large Scale integrated circuits available to the numercal analyst involved in structural engineering computations. Processor organization, including parallel and pipeline processors, is examined and current limitations noted. VLSI hardware is examined in three phases: general purpose chips, semi-custom chips, and custom purpose chips. Specific topics include: read-only memories, micro-programmable chips, programmable logic arrays, and CAD tools used to design integrated circuits. An appendix discusses

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

the manufacturing aspects of integrated circuits. A glossary of applicable terminology is included and ain index is also provided.